９０１画像処理ライブラリ
FVL基本SDK/902(FVL-SDK-BSC/902)
９０３画像処理ライブラリ
FVL基本SDK/904(FVL-SDK-BSC/904)
FVL基本SDK/DOS(FVL-SDK-BSC/DOS)
FVL基本SDK/LNX(FVL-SDK-BSC/LNX)

90X
濃淡画像ライブラリ説明書

☆第１５版☆

（株）ファースト

はしがき

この説明書は、

９０１画像処理ライブラリ　Ver.6.50

ＦＶＬ基本ＳＤＫ／９０２(902画像処理ﾗｲﾌﾞﾗﾘ)　Ver.3.40

９０３画像処理ライブラリ　Ver.1.40

ＦＶＬ基本ＳＤＫ／９０４(904画像処理ﾗｲﾌﾞﾗﾘ)　Ver.1.30

ＦＶＬ基本ＳＤＫ／ＤＯＳ　Ver.3.40

ＦＶＬ基本ＳＤＫ／ＬＮＸ　Ver.1.80

以降に対応しています。

本書は、９０Ｘシリーズ用ユーザプログラム作成時に使用できる、「濃淡画像ライブラリ」について記載したものです。

なお、この他にＣＳＣ９０Ｘシリーズライブラリとしては、下記のようなものがあります。

◎９０Ｘ　基本ライブラリ（Ｖｏｌ．１，Ｖｏｌ．２，Ｖｏｌ．３）説明書

◎９０Ｘ　２値画像ライブラリ説明書

◎９０Ｘ　ビジョン・ツール・ライブラリ説明書

◎９０Ｘ　キャリパーライブラリ説明書

ユーザプログラムの開発方法その他につきましては

◎FAST Vision Libraryプログラマーズガイド

◎９０Ｘ　操作説明書

◎FVL/LNX　操作説明書

をご参照ください。

［注１］９０１用のコンパイラ ＭＣＰ９６０（９６０　Ｃクロスコンパイラ）のバージョンは２．０以上です。

［注２］９０２用のコンパイラ Ｗａｔｃｏｍ　Ｃのバージョンは１０．５以上です。

［注３］９０３，９０４用のコンパイラ Green Hills C および MULTIのバージョンは１．８．９以上です。

目　　次

目　　次

目　　次

1. 概　　要

本書はＣＳＣ９０Ｘシリーズ　濃淡画像ライブラリについて、記載します。

①　グレイサーチライブラリ

②　Ｓ回転サーチライブラリ

③　直線検出ハフ変換ライブラリ

④　新直線検出ハフ変換ライブラリ

⑤　エッジサーチライブラリ

⑥　濃淡エッジ計測ライブラリ

⑦　画像強調・フィルタリングライブラリ

⑧　メモリ間転送・演算ライブラリ

⑦　濃度変換ライブラリ

⑧　画像計測ライブラリ

なお、各ライブラリ関数の表記

９０１	９０２	９０３	９０４	FVL/LNX	高分解能

は、９０１，９０２，９０３，９０４及び高分解能カメラ使用時にライブラリ関数が使用できるかどうかを表すもので

○　・・・・・・・・・・　使用可能
△　・・・・・・・・・・　一部使用可能
空白・・・・・・・・・・　使用不可

となっております。
△の場合は、必ず留意事項を参照してください。

2．グレイサーチライブラリ

本ライブラリはグレイ画像メモリの中から、あらかじめ登録しておいたサーチ・パタンの位置を探し出したり、類似性を計測するものです。
サーチ処理方式は、グレイレベルにて濃度を正規化したパタンマッチングを行っていますので

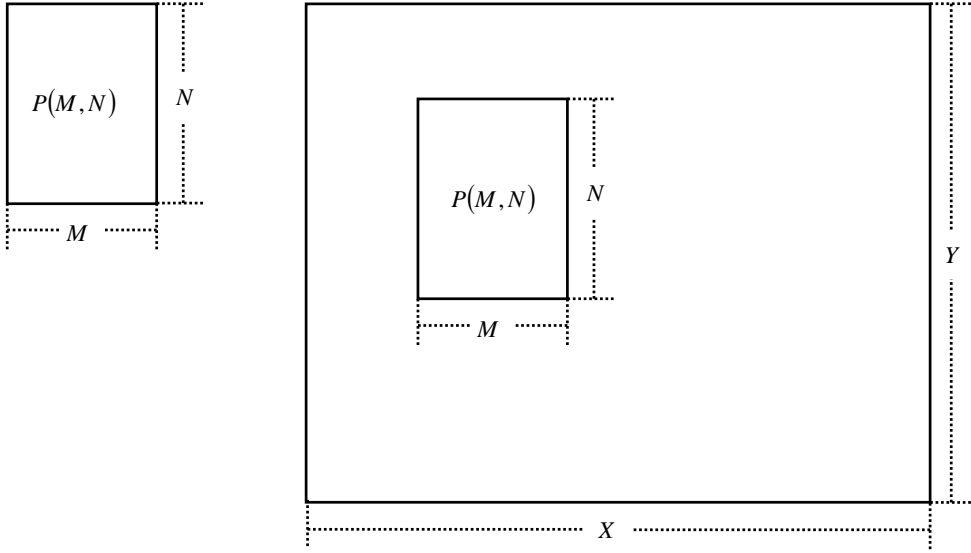
・外乱光及び照明装置の劣化等による、照度の変化
・ＩＴＶカメラのフォーカスのズレ
・サーチするパタンの変化（小量のキズ、カケ等がある場合）

に影響されることが少なく、２値画像では認識できない複雑な階調を持ったパタンを、探し出すことができます。
サーチに要する時間は、サーチ・パタンのサイズ、サーチするエリアの大小や、サーチ精度、サーチ方式によって左右されますが、概ね巻末の「グレイサーチ処理時間の条件」の様に高速度で実行されます。
しかし、巻末の「グレイサーチ処理時間の条件」からもわかるように

・サーチエリアが大きい場合･･･ 探す領域が増える
・高精度の場合･･･････････････ 位置を抽出するための計算量が増える

には、当然のごとく速度は低下します。

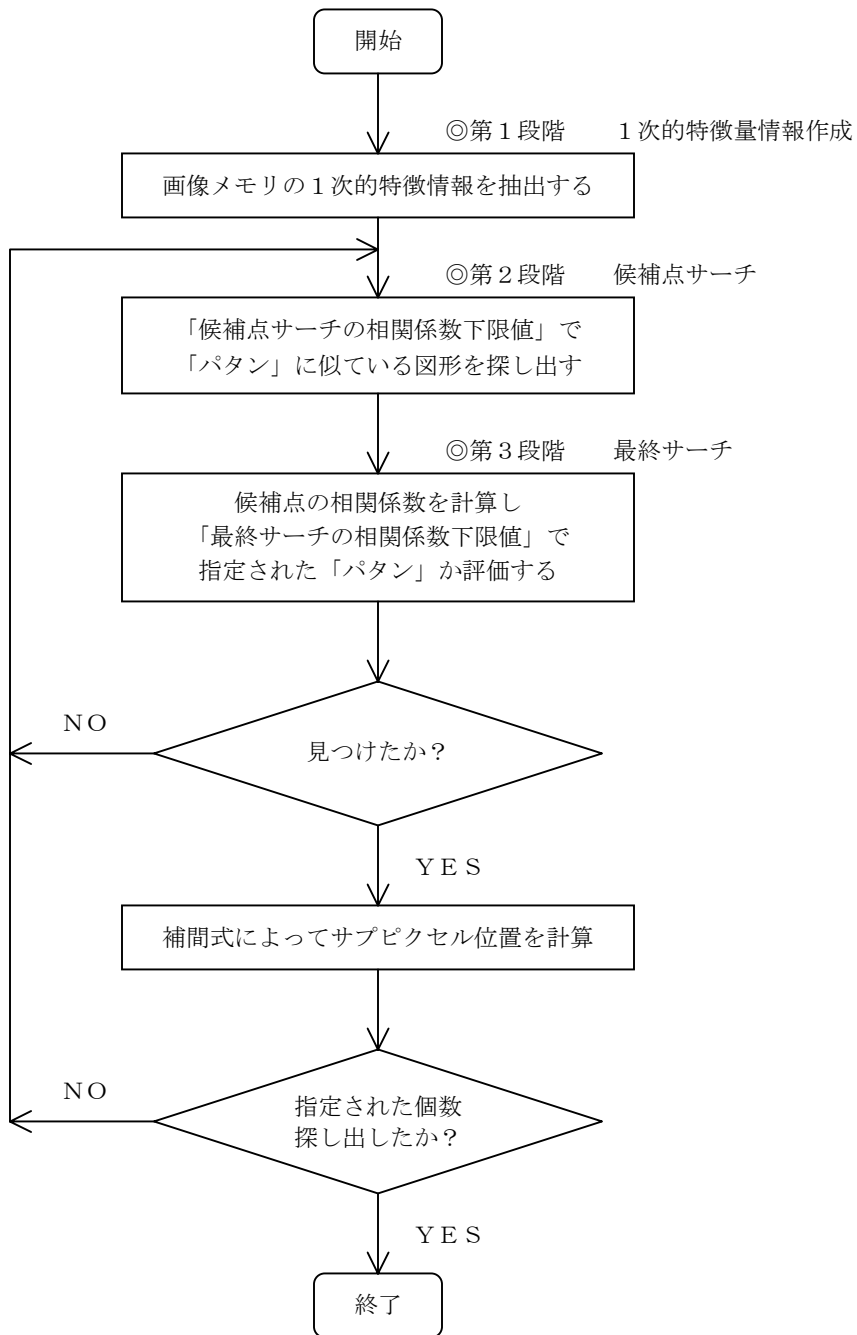
2.1　サーチ処理アルゴリズム

サーチするパタンの画像 $P(M,N)$ と、サーチエリアの画像 $B(X,Y)$ の部分画像 $B_{ij}(M,N)$ との相互相関係数 C_{ij} （マッチングの度合を示す情報）を次式の計算で求めて、その値がパラメータ指定値より大きくなった座標値を出力します。

$$C_{ij} = \frac{M \cdot N \cdot \left(\sum_{m=1}^{M} \sum_{n=1}^{N} P(m,n) B_{ij}(m,n) \right) - \left(\sum_{m=1}^{M} \sum_{n=1}^{N} P(m,n) \right) \cdot \left(\sum_{m=1}^{M} \sum_{n=1}^{N} B_{ij}(m,n) \right)}{\sqrt{\left\{ M \cdot N \cdot \sum_{m=1}^{M} \sum_{n=1}^{N} P(m,n)^2 - \left(\sum_{m=1}^{M} \sum_{n=1}^{N} P(m,n) \right)^2 \right\} \cdot \left\{ M \cdot N \cdot \sum_{m=1}^{M} \sum_{n=1}^{N} B_{ij}(m,n)^2 - \left(\sum_{m=1}^{M} \sum_{n=1}^{N} B_{ij}(m,n) \right)^2 \right\}}}$$

C_{ij} は常に $-1 \leqq C_{ij} \leqq 1$ の値となり、特に $C_{ij}=1$ の場合はパタン画像 $P(m,n)$ と部分画像 $B_{ij}(m,n)$ が完全に一致している事を示します。

本ライブラリではサーチのしきい値（パタンを見つけたと判断する値）として、この C_{ij} （以降「相関係数」と呼ぶ）を10000倍した値をパラメータとして指定します。

（１）サーチ処理手順

サーチ処理は下図のように３段階で行われます。

［注］サーチ精度によって計算方法が異なる。

（2）サーチ処理とパラメータとの関係

弊社のグレイサーチはサーチ速度を向上させるため、マスタパタンと入力画像の情報を数段階に圧縮し、圧縮レベルの高い画像から低い画像へ圧縮レベルを落としながらサーチを行います。
そのため、入力画像を見る限り問題なくサーチできそうな画像に対しても、サーチできない、間違った位置を出力する、といった動作をすることがあります。
この問題を回避するためにいくつかのパラメータを用意しておりますので、適切に設定してお使いください。

◆ パタン登録時のパラメータについて

・特殊サーチ制御モード

「特殊サーチ制御モード」パラメータでは、マスタパタンと入力画像の圧縮方法を変更できます。
細い線が含まれる対象物(文字など)では、画像を圧縮する過程で情報が消えてしまうことがあるので、Lib_gs_smode() 関数で圧縮方法を変更する必要があります。
線の部分の濃度レベルが背景部分よりも低ければ「BLACK_LINE_PATTERN」を、逆であれば「WHITE_LINE_PATTERN」を選択してください。

◆ サーチ実行時のパラメータについて

・複雑度

「複雑度」パラメータはサーチを開始する圧縮レベルを決定します。
設定値が大きくなるほど低い圧縮レベルからサーチを始めるため、サーチ処理の時間が大きくなりますが、圧縮の影響で対象物を見失う確率が低くなります。

・精度

「精度」パラメータはどの圧縮レベルで最終評価をするかを決定します。
「通常精度」→「高精度」→「超高精度」の順で最終サーチを行う圧縮レベルが低くなり、「超高精度」では最終サーチを情報圧縮なしで行います。
基本的にはサーチ結果に必要とされる精度によって設定値を選択して頂きますが、入力画像の中で本来のサーチ対象物の他に似たような(相関値が高い)対象物がある場合は、誤検出の可能性を低くするためにより高い精度を選択してください。

・途中下限値

候補点サーチ（注 1）では、「途中下限値」パラメータの設定値以上の相関値が得られた点のみ、圧縮レベルを低くしながら繰り返し評価されます。
設定値を低くすると評価する点が多くなるため、サーチ処理の時間が大きくなりますが、圧縮の影響で対象物を見失う確率が低くなります。
「複雑度」パラメータと似ていますが、一般的にはサーチ処理時間の増大量が「途中下限値」パラメータを低くした方が大きくなります。

・最終下限値

最終サーチ（注 2）では、「最終下限値」パラメータの設定値以上の相関値が得られた点のみ、回答として報告されます。
設定値を低くすると相関値の低い対象物も回答として報告されるようになるので、対象物に多少の変化(一部の欠け、多少の回転など)があっても見つけることがでますが、意図していた対象物以外の回答が混在する可能性が高くなります。

（注 1）「複雑度」パラメータで決定されるサーチ開始圧縮レベルから「精度」パラメータで決定されるサーチ終了圧縮レベルまでのサーチを候補点サーチと呼んでいます。
（注 2）サーチ終了圧縮レベルでのサーチを最終サーチと呼んでいます。

（3）各パラメータの決定方法について

◆ マスタパタンの登録

・マスタパタンの登録では、なるべく探したいパタンの背景部分が十分に含まれるようにサイズを指定してください。

弊社のグレイサーチでは明るさの変化を吸収してパタンを見つけることができます。その反面、何もないと思われるような背景領域でも微小な濃度の変化を検出して、比較的高い相関値を出力することがあります。

これを軽減するには、マスタパタンに濃度変化の大きい部分と濃度変化の小さい部分が同程度含まれていることが必要です。

例えば「A」という文字をマスタパタンとする場合は、文字の周囲の余白部分も含めて登録すれば何もない背景領域を回答として出力する確率が低くなります。

・マスタパタンに細い線が含まれる場合は、「特殊サーチ制御モード」パラメータを「BLACK_LINE_PATTERN」または「WHITE_LINE_PATTERN」に設定してください。

複雑度を９、途中下限値を 1000、最終下限値を 6000 に設定してもパタンが検出できない場合は、圧縮の影響でマスタパタンの情報が消えてしまっている可能性があります。

この場合は、マスタパタン登録時に「特殊サーチ制御モード」パラメータを設定してください。

◆ 精度、最終下限値の決定

・「精度」、「最終下限値」パラメータは、対象物の変化(欠け、回転など)による相関値の低下と第２候補(本来の対象物の次に相関値が高い対象物)の相関値を検証した上で決定してください。

マスタパタンを登録したら、最初に「精度」、「最終下限値」パラメータを決定します。

この時点では圧縮による影響を最低限に抑えるため、「複雑度」、「途中下限値」パラメータをそれぞれ、９、1000 に設定してください。

その後、対象物の変化に対応でき誤検出のない「精度」、「最終下限値」パラメータを以下の手順で探していきます。

① 「精度」パラメータを、サーチ結果に必要とされる精度によって選択してください。
② 「サーチ個数」パラメータを、検出したい対象物の数＋１に設定してください。
③ 「最終下限値」パラメータを 5000 に仮設定します。
④ この設定で実際に対象物の画像を取り込み、対象物の変化に対応できるかどうか評価してください。対象物が撮像されてない場合や、登録位置からずらした場合などもあわせて評価します。
⑤ ④の評価で検出したい対象物以外の対象物が検出されている場合は、最終相関値を検出したい対象物以外の対象物の相関値より大きく設定して、再度④の評価を実施します。
　これを誤検出がなくなるまで繰り返してください。
⑥ ④、⑤を繰り返しても誤検出がなくならない場合、検出できない対象物がある場合は「精度」パラメータを「超高精度」に変更します。
⑦ 誤検出がなくなり、全ての対象物を検出できれば「精度」、「最終下限値」パラメータは決定です。

上記手順で誤検出がなくならない場合、または検出できない対象物がある場合はマスタパタンの変更を検討してください。それでも問題がある場合は、弊社のグレイサーチでは安定してサーチできないということになります。

◆ 複雑度、途中下限値の決定

・「複雑度」パラメータは、処理時間の許す限り高く、「途中下限値」パラメータは、低く設定してください。

「複雑度」、「途中下限値」パラメータは処理時間と対象物の検出確率に影響を与えます。多くのケースにおいて「複雑度」パラメータを９に固定して、「途中下限値」パラメータを 1000 から上昇させていき、処理時間が許容範囲に入ったところで決定する方法が最善だと思われます。

すべてのパラメータ決定後、誤検出、検出できない対象物、処理時間などを再度評価してみてください。

Lib_gs_defadrs

機　能

サーチ・パタン定義エリア指定

形　式

```
#include "f_search.h"
int Lib_gs_defadrs( char *p_adrs, int p_size, int p_mode );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	△

解　説

サーチの対象となるパタンを登録しておくメインメモリの領域を設定します。
本ライブラリでは最大２００個のサーチ・パタン（異なるサイズのパタンも登録可能）を、登録（管理）することができます。
本ライブラリは、以降のグレイサーチ・ライブラリを引用する前に必ず一度コールして、出力パラメータである関数値がゼロにならなければなりません。

① ***p_adrs** はサーチの対象となるパタンを登録するメインメモリ領域の先頭アドレスを指定します。

② **p_size** はユーザが確保した格納領域のサイズを、バイト数換算値で指定します。
登録する各サーチ・パタンのサイズ（＝横方向画素数＊縦方向画素数）を

$N_i(i=0...199)$ とすると

(90Xをお使いの場合)

$$1024+(N_0*1.34+950)+(N_1*1.34+950)+\cdots\cdots+(N_i*1.34+950)$$

(FVL/LNXをお使いの場合)

$$256+(N_0*1.50+848)+(N_1*1.50+848)+\cdots\cdots+(N_i*1.50+848)$$

の領域が必要となります。

③ **p_mode** は初期化モードを指定します。

値	定　　数	意　　味
0	INITIAL_PTN_AREA	格納エリアを初期化する場合。
1	CONTINUE_PTN_AREA	格納エリアを継続して使用する場合。

戻り値

処理結果

値	意　　味
0	正常終了しました。
－1	格納領域サイズ不足(登録最小パタンが、１個も格納できない)です。
－2	格納領域が所定の形式になっていません。 (p_mode に CONTINUE_PTN_AREA を指定した場合のみ)

例

```
#include "f_search.h"
#include "f_stdlib.h"

#define GRAY_PTN_SIZE   0x10000

int ptn_init()
{
    char *gray_adrs;
    int  rtn_code;

    rtn_code = -1;

    /*ﾒﾓﾘの確保*/
    if( NULL != ( gray_adrs = Lib_mlalloc( GRAY_PTN_SIZE )))
    {
        if ( 0 == Lib_gs_defadrs( gray_adrs, GRAY_PTN_SIZE, INITIAL_PTN_AREA ))
            rtn_code = 0;
    }
    return( rtn_code );
}
```

留意事項

○かつて一度も使用したことのない領域を p_adrs で指定した場合は p_mode で INITIAL_PTN_AREA を指定します。

○サーチパタンのサイズが512×256画素まで対応しています。それ以上のサーチパタンサイズに対しては、サーチ・パタン定義エリア指定Ⅱ(Lib_gs_xdefadrs)を実行し、8bit版グレイサーチを使用してください。

90X

○サーチ・パタン格納領域の形式

サーチ・パタンデータの格納形式を下図に示します。（先頭アドレスを P_0 とした場合）

［サーチ・パタンデータ格納形式（90Xの場合）］

アドレス	内容
P_0＋０	ライブラリ使用エリア
P_0＋４	登録済みサーチ・パタン個数（０～２００）
P_0＋８	エリアサイズ（バイト数換算値）
P_0＋１２	「サーチ・ウインドウ」ｘ始端位置（初期値＝０）
P_0＋１６	「サーチ・ウインドウ」Ｙ始端位置（初期値＝０）
P_0＋２０	「サーチ・ウインドウ」ｘ終端位置（初期値＝５１１）
P_0＋２４	「サーチ・ウインドウ」Ｙ終端位置（初期値＝４７９）
P_0＋２８	候補点サーチの相関係数下限値（初期値＝５０００）
P_0＋３２	最終サーチの相関係数下限値（初期値＝６０００）
P_0＋３６	サーチエリアの複雑度（１～９，初期値＝５）
P_0＋４０	ウインドウ接触パタン検出有無（初期値＝０：除去）
P_0＋４４	隣接パタンＸ方向画素数（＝初期値０：パタンの１／２）
P_0＋４８	隣接パタンＹ方向画素数（＝初期値０：パタンの１／２）
P_0＋５２ ・ P_0+127	Ｒｅｓｅｒｖｅｄ
P_0+128	１番目のサーチ・パタンの格納アドレス＝A_1
	２番目のサーチ・パタンの格納アドレス＝A_2
・	・　　・
P_0+924	２００番目のサーチ・パタンの格納アドレス＝A_{200}
P_0+928 ・	Ｒｅｓｅｒｖｅｄ
P_0+1023	
A_1	１番目のサーチ・パタンのデータ（詳細は次頁※参照）
A_2	２番目のサーチ・パタンのデータ
A_3 ・ A_{200}	３～２００番目のサーチ・パタンのデータ

［注１］この部分には、アドレス情報（絶対番地）等が記録されているため、内容を変更すると、グレイサーチライブラリの正常動作は保証されません。

［注２］画像イメージデータは６ビット画像に圧縮されて格納されます。
（最上位ビットはマスクデータとして使用されます）

FVL/LNX

○90X とサーチパタンの格納形式が異なるため、90X のサーチパタンファイルを FVL/LNX で使用することは出来ません。
また、サーチパタンの登録個数の制限はありません。

○サーチパタンに格納されている画像イメージデータは８ビット画像です。（90X では６ビット画像に圧縮されています。）

○FVL/LNX のサーチパタン格納形式は将来変更の可能性があるため、公開しておりません。

Lib_gs_xdefadrs

機　能

サーチ・パタン定義エリア指定Ⅱ

形　式

```
#include "f_search.h"
int Lib_gs_xdefadrs( char *p_adrs, int p_size, int p_mode );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	◯			◯	◯

既存のグレイサーチライブラリは 1000×1000 などの大きなサイズのパタンを登録し、サーチを実行した場合に、Ｘ，Ｙ座標値，相関値の結果が正しく計測できない場合があります。
（ラインセンサカメラなど画像サイズの大きいカメラを使用した場合に上記現象が発生します。）
この問題に対応するため、内部演算を 64bit に拡張したライブラリが「８ｂｉｔ版グレイサーチ」です。
既存のグレイサーチとの互換性に関する詳細は後述の留意事項を参照してください。
８ｂｉｔ版グレイサーチは既存の Lib_gs_defadrs の代わりに Lib_gs_xdefadrs をコールすれば実行されるようになっております。（なお FVL/LNX の場合は、Lib_gs_defadrs・Lib_gs_xdefadrs どちらをコールしても８ｂｉｔ版が実行されます。）
ラインセンサカメラなど画像サイズの大きいカメラを使用する場合には、８ｂｉｔ版グレイサーチを使用してください。

解　説

サーチの対象となるパタンを登録しておくメインメモリの領域を設定します。
本ライブラリでは最大２００個のサーチ・パタン（異なるサイズのパタンも登録可能）を、登録（管理）することができます。
本ライブラリは、以降のグレイサーチ・ライブラリを引用する前に必ず一度コールして、出力パラメータである関数値がゼロにならなければなりません。

① ***p_adrs** はサーチの対象となるパタンを登録するメインメモリ領域の先頭アドレスを指定します。

② **p_size** はユーザが確保した格納領域のサイズを、バイト数換算値で指定します。
登録する各サーチ・パタンのサイズ（＝横方向画素数＊縦方向画素数）を
$N_i \left(i = 0 \ldots 199 \right)$ とすると

(90Xをお使いの場合)

$$1024 + \left(N_0 * 1.34 + 950\right) + \left(N_1 * 1.34 + 950\right) + \cdots\cdots + \left(N_i * 1.34 + 950\right)$$

(FVL/LNXをお使いの場合)

$$256 + \left(N_0 * 1.50 + 848\right) + \left(N_1 * 1.50 + 848\right) + \cdots\cdots + \left(N_i * 1.50 + 848\right)$$

の領域が必要となります。

③ **p_mode** は初期化モードを指定します。

値	定　　数	意　　味
0	INITIAL_PTN_AREA	格納エリアを初期化する場合。
1	CONTINUE_PTN_AREA	格納エリアを継続して使用する場合。

※Lib_gs_defadrsをコールして作成されたパタンファイルに対し、本関数で継続使用(CONTINUE_PTN_AREA)を指定した場合については留意事項をご覧下さい。

戻り値

処理結果

値	意　　　　味
0	正常終了しました。
－1	格納領域サイズ不足(登録最小パタンが、１個も格納できない)です。
－2	格納領域が所定の形式になっていません。 (p_mode に CONTINUE_PTN_AREA を指定した場合のみ)

例

```
#include "f_search.h"
#include "f_stdlib.h"

#define GRAY_PTN_SIZE   0x10000

int ptn_init()
{
    char *gray_adrs;
    int  rtn_code;

    rtn_code = -1;

    /*ﾒﾓﾘの確保*/
    if( NULL != ( gray_adrs = Lib_mlalloc( GRAY_PTN_SIZE )))
    {
        if ( 0 == Lib_gs_defadrs( gray_adrs, GRAY_PTN_SIZE, INITIAL_PTN_AREA ))
            rtn_code = 0;
    }
    return( rtn_code );
}
```

留意事項

○従来のグレイサーチライブラリ(Lib_gs_defadrs)ではパタン画像を 6bit に圧縮していましたが、8bit 版グレイサーチ(Lib_gs_xdefadrs)では 8bit のまま使用しています。よって、従来より濃淡変化の小さい対象物もサーチ可能になっております。

90X

○ラインセンサカメラを使用する場合でも 1000×1000 などの大きなサイズのパタンを登録しない場合(サーチパタンのサイズが 512×256 画素以下の場合)、従来のグレイサーチライブラリ(Lib_gs_defadrs)を使用しても問題はありません。

○Lib_gs_defadrs をコールして作成されたパタンファイルに対し、本関数で継続使用を指定した場合、エラーが返ります。(6bit 版は Lib_gs_defadrs を使用して下さい。)

FVL/LNX

○Lib_gs_defadrs・Lib_gs_xdefadrs どちらを使用した場合も 8bit 版として動作します。

○Lib_gs_defadrs をコールして作成されたパタンファイルに対し、本関数で継続使用を指定した場合、そのままお使い頂けます。(Lib_gs_defadrs を使用した場合も 8bit 版として動作します。)

Lib_gs_defpat

機　能

サーチ・パタン定義

形　式

```
#include "f_search.h"
int Lib_gs_defpat ( int p_name );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	

解　説

モニタＴＶ表示を見ながらマウスを操作して、グレイメモリ上の画像データをサーチの対象となる「サーチ・パタン」に登録します。

① ***p_name*** はパタン名称です。名称は４文字のｎｕｌｌコード以外を付けてください。
（登録するサーチ・パタンに付与する名前）
［注］ここで設定した名称で、サーチ実行(=Lib_gs_search)ライブラリでサーチするパタンを指示します。

戻り値

処理結果

値	意　　　味
０	登録が正常終了しました。
－１	既に２００個登録済みのため登録ができません。
－２	サーチ・パタン格納領域が残りすくないために登録ができません。（登録済み個数は、まだ２００個未満）
-10	パタン名称にnullコードが指定されました。
1	マウス操作のキャンセル指示により、登録処理を取り消しました。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

設定方法

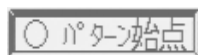
映像データに重畳して、下記の項目がグラフィック表示されます。

・登録する画像範囲‥‥‥‥ ボックスカーサ

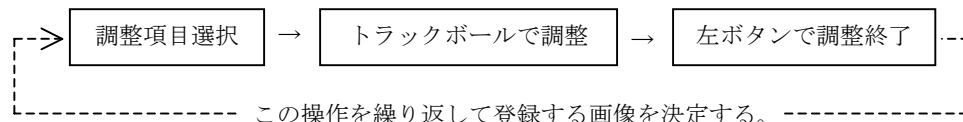
・サーチ結果通知位置‥‥‥ クロスカーサ

以下の項目を操作してサーチパタンの画像を登録します。

［注］すでに同じ名称のパタンが登録されている場合は、パッドタイトルの表示が パタン更新 となります。

中心位置 (Center)
サーチ結果として通知する座標位置を調整します。

パターン始点 (Ptn start)
サーチパタンとして登録する画像の左上端位置を調整します。

パターンサイズ (Ptn size)
サーチパタンとして登録する画像の大きさを調整します。（調整単位は２画素です）

上記の項目は、選択するとトラックボールの操作により調整できる機能が変わります。
選択した項目の調整は「左ボタン」を押すと終了となります。

実 行 (EXEC)
サーチパタンを登録します。
（メモリに記憶されるだけでファイル装置には記憶されません。）

取消 (CANC)
サーチパタンを登録しません。操作ミスでこの画面になった場合にご使用ください。

例

現在入力中の画像データより、サーチ処理の基準となるパタンを作成します。
パタンとして登録する画像の位置・サイズはマウスを操作して指定します。

```
#include "f_search.h"

int define_ptn_data()
{
    int name;

    name = 0x44434241;   /*ABCD*/

    return( Lib_gs_defpat( name ));
}
```

留意事項

○登録可能な「サーチ・パタン」の大きさ及び形状
　登録できる「サーチ・パタン」は、幅・高さが偶数で面積が 65536画素以内の矩形に限定されます。

○一定濃度のサーチ・パタンは登録できません。
　本ライブラリのサーチ処理は、前述の“２．サーチ処理アルゴリズム”に記載してある「相互相関係数」を使用していますので、濃度レベルが全面に渡って均一のパタンを、サーチ・パタンとして登録することはできません。

○「サーチ・パタン」の更新
　登録サーチ・パタン名称として、既に登録済みの名称を指定した場合には、「置換」指示であるとみなされて新たな画像イメージデータでオーバライトされます。

○登録可能な「サーチパタンの位置」
　登録位置はなるべく画面中央で行ってください。
　「サーチパタン」の外側（４方向）に２０画素の余裕が必要です。

○高分解能カメラを使用する場合は Lib_gs_usepat または Lib_gs_exdefpat を使用してください。

Lib_gs_defmask

機　能

マスクパタン設定

形　式

```
#include "f_search.h"
int Lib_gs_defmask ( int p_name );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

モニタＴＶ表示を見ながらマウスを操作して、登録済みの「サーチ・パタン」に対する不感帯マスク（＝サーチの対象としない部分）の設定を行います。
既に、マスクが設定されている場合はマスクの修正を行うこととなります。

① ***p_name*** はマスクの設定・修正を行うパタン名称です。
　パタン名称は Lib_gs_defpat, Lib_gs_usepat ライブラリで登録したものを使用します。

戻り値

処理結果

値	意　　　味
０	マスク設定が正常終了しました。
－１	パラメータで指定された名称の「サーチ・パタン」は登録されていないので処理ができません。
１	マウス操作のキャンセル指示により、処理を取り消しました。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

設定方法

マスク（サーチするときの不感帯）の、設定・解除を行います。
モニタＴＶには登録されているパタンの画像データとともに、マスクが設定されている部分が“黒”（濃度レベル：０）で表示されます。

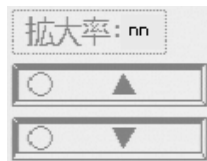
パタンの画像範囲は、ボックスカーサでグラフィック表示されます。 ➡

(Move)
ここを選択すると、トラックボールにより「パタンの表示位置」を移動できます。
位置の確定は、左または右ボタンを押します。

(Mask ON)

(Mask OFF)
実際のマスク調整操作を行います。
ここを選択すると、「左ボタン」を押す毎にカーソル位置のパタン画像の、マスク設定（マスク・オンを選択時）またはマスク解除（マスク・オフを選択時）が行われます。
「右ボタン」を押すと調整操作の終了となります。

(All CLEAR)
現在設定されているマスクを全て解除します。
選択すると、確認用の「オーバレイパッド」が表示されます。

(Zone)
領域マスクの設定・解除操作に移ります。

(Zoom Rate)

パタンは拡大表示することができます。ｎｎは現在の表示倍率です。
［▲］と［▼］を選択すると表示倍率が変更できます。

(Pen size)

マスク調整操作で、「左ボタン」を１回押したときに変更されるマスクの画素数を「ペンサイズ」と呼びます。ｍｍは現在のペンサイズです。［▲］と［▼］で変更できます。

例

既に登録済みの“ＡＢＣＤ”なる名称の「サーチ・パタン」のマスクを設定（修正）します。

```
#include "f_search.h"

int set_mask()
{
    int name;

    name = 0x44434241;    /*ABCD*/

    return( Lib_gs_defmask( name ));
}
```

留意事項

○マスク設定不可の場合即リターン
パラメータで指定した名称の「サーチ・パタン」が未登録の場合は、出力パラメータの関数値にー１をセットして、ユーザプログラムに戻ります。

○不感帯マスクのパーセンテージ
マスクの設定されない部分が１０００画素程度（連続した領域の必要あり）残れば、任意の大きさのマスクを設定することができます。（「サーチ・パタン」の何パーセント迄ということではありません）

Lib_gs_usepat

機　能

ユーザ指定サーチ・パタン登録

形　式

```
#include "f_search.h"
int Lib_gs_usepat( int p_name, int p_sx, int p_sy, int p_xsize,
                    int p_ysize, int p_xofset, int p_yofset );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

グレイメモリの画像データを、サーチの対象となる「サーチ・パタン」に登録します。
サーチ・パタン定義(=Lib_gs_defpat)ライブラリでは、モニタＴＶ表示を見ながら、マウスを操作して、登録するパタンのサイズ・グレイメモリ上の画像位置を指定しましたが、本ライブラリでは、それらをライブラリ引用時のパラメータ値で直接指定します。

① ***p_nam*** はパタン名称です。名称は４文字のｎｕｌｌコード以外を付けてください。
（登録するサーチ・パタンに付与する名前）
［注］ここで設定した名称で、サーチ実行(=Lib_gs_search)ライブラリでサーチするパタンを、指示します。

② ***p_sx*** は「サーチ・パタン」に登録する画像の左上端Ｘ座標です。
範囲は ２０≦ p_sx ＜ (横方向画像サイズ) － p_xsize －２０ の数値で指定します。

③ ***p_sy*** は「サーチ・パタン」に登録する画像の左上端Ｙ座標です。
範囲は ２０≦ p_sy ＜ (縦方向画像サイズ) － p_ysize －２０ の数値で指定します。

④ ***p_xsize*** は登録する「サーチ・パタン」の横方向画素数です。
範囲は ２≦ p_xsize ≦ (横方向画像サイズ) － p_sx －４０ のうちの、偶数値で指定します。

⑤ ***p_ysiz*** は登録する「サーチ・パタン」の縦方向画素数です。
範囲は ２≦ p_ysize ≦ (縦方向画像サイズ) － p_sy －４０ のうちの、偶数値で指定します。

⑥ ***p_xofset*** はサーチ実行時に返答されるＸ座標位置のオフセットです。
“Ｘ座標”は、「サーチ・パタン」の左上端に、ここで指定したオフセットを加算した座標位置となります。
指定可能な範囲は －９９９９９～＋９９９９９ です。（単位：０．１画素）

⑦ ***p_yofset*** はサーチ実行時に返答されるＹ座標位置のオフセットです。
“Ｙ座標”は、「サーチ・パタン」の左上端に、ここで指定したこのオフセットを加算した座標位置となります。
指定可能な範囲は －９９９９９～＋９９９９９ です。（単位：０．１画素）

戻り値

処理結果

値	意　　　　味
０	登録が正常終了しました。
－１	既に２００個登録済みのため登録ができません。
－２	サーチ・パタン格納領域が残りすくないために登録ができません。 （登録済み個数は、まだ２００個未満）
－３	パラメータ・エラー（始端Ｘ・Ｙ位置が範囲外）です。
－４	パラメータ・エラー（パタンＸ・Ｙ方向サイズが範囲外）です。
－５	パラメータ・エラー （センター・マークＸ・Ｙ方向オフセットが範囲外）です。
－６	パラメータ・エラーです。 （始端Ｘ・Ｙ位置とパタンＸ・Ｙ方向サイズとの関係が範囲外）
－７	指定された画像データは一定レベルのため登録不可 （サーチできない）
-10	パタン名称に null コードが指定されました。
-999	サーチ･パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

グレイメモリ上の、左上端座標がＸ＝２０・Ｙ＝２０で、１辺が９６画素の正方形の画像データを“ＡＢＣＤ”なる名称で「サーチ・パタン」に登録します。
サーチ実行時には、パタンの左上端にＸ方向に３５．０，Ｙ方向に２５．５加算した点に対応する座標位置を返答します。

```
#include "f_search.h"

int set_ptn_data()
{
    int name;

    name = 0x44434241;    /*ABCD*/

    return( Lib_gs_usepat( name, 20, 20, 96, 96, 350, 255 ));
}
```

留意事項

○登録可能な「サーチ・パタン」の大きさ及び形状登録できる「サーチ・パタン」は、幅・高さが偶数で面積が(横方向画像サイズ) ×(縦方向画像サイズ)／４画素以内の矩形に限定されます。
（マスク設定により、サーチ・パタンの有効部分を任意の形状にすることができます)

○一定濃度のサーチ・パタンは登録不可
本ライブラリのサーチ処理は、前述の“２．サーチ処理アルゴリズム”に記載してある「相互相関係数」を使用していますので、濃度レベルが均一な「サーチ・パタン」を登録することはできません。

○マスクの状態
本ライブラリで登録時は「マスクは全て無し」となります。

○「サーチ・パタン」の更新
登録サーチ・パタン名称として、既に登録済みの名称を指定した場合は新たな画像イメージデータでオーバライトされます。

○グレイメモリ以外の領域にあるデータを登録する場合
登録したいパタンの画像イメージを、一旦グレイメモリに展開する必要があります。
この場合“画像イメージデータ”は８ビット濃淡画像でなければなりません。
（特に、「サーチ・パタン」格納領域に登録済みの、「サーチ・パタン」の画像データを参照する場合等は、６ビットに圧縮されていますので注意してください)

○登録可能な「サーチパタンの位置」
登録位置はなるべく画面中央で行ってください。
「サーチパタン」の外側（４方向）に２０画素の余裕が必要です。

Lib_gs_fremask

機　能

自由形状マスク登録

形　式

```
#include "f_search.h"
int Lib_gs_fremask ( int p_name, char *mask_data );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

パラメータで指定される名称の「サーチ・パタン」に対して、ユーザが指定する任意形状の不感帯マスク（＝サーチの対象としない部分）の設定を行います。
既に、マスクが設定されている場合はマスクの修正（置換）を行うこととなります。

① ***p_name*** はマスクの設定・置換を行うパタン名称です。
　パタン名称は Lib_gs_defpat, Lib_gs_usepat ライブラリで登録したものを使用します。

② ****mask_data*** はマスクデータが格納されている領域の先頭アドレスです。
　　　１：マスクをセットします
　　　０：マスクしません

戻り値

処理結果

値	意　　　味
０	マスク設定が正常終了しました。
－１	パラメータで指定された名称の「サーチ・パタン」は登録されていないので処理ができません。
－２	マスク部分が多すぎてサーチが実行できなくなってしまうので、指定したマスクは設定できません。（元のマスクのままで変更されない）
-999	サーチ･パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

パタンサイズが、横６４画素、縦１２８画素の“ＡＢＣＤ”なる名称の「サーチ・パタン」を登録後、マスクを設定します。

```
#include "f_search.h"

int set_mask()
{
    int name;
    int rtn_code;
    static char mask_buf[64*128];

    name = 0x44434241;   /*ABCD*/

    if( 0 == (rtn_code = Lib_gs_usepat( name, 10, 20, 64, 128, 350, 255 )) )
    {
        /* mask_buf[] に任意形状のﾏｽｸﾃﾞｰﾀを作成する。*/

            return( Lib_gs_fremask( name, mask_buf ));
    }
    else
         /*ｴﾗｰ処理*/;
}
```

留意事項

○不感帯マスクのパーセンテージ

マスクの設定されない部分が１０００画素程度（連続した領域の必要あり）残れば、任意の大きさのマスクを設定することができます。（「サーチ・パタン」の何パーセント迄ということではありません）

Lib_gs_dsppat

機　能

登録サーチ・パタン表示（削除）

形　式

```
#include "f_search.h"
int Lib_gs_dsppat ( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

登録されている「サーチ・パタン」の画像イメージをモニタＴＶに表示したり、削除を行います。
表示・削除する「サーチ・パタン」はマウスにより１個ずつ選択します。

戻り値

処理結果

値	意　　　味
0	正常終了しました。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

設定方法

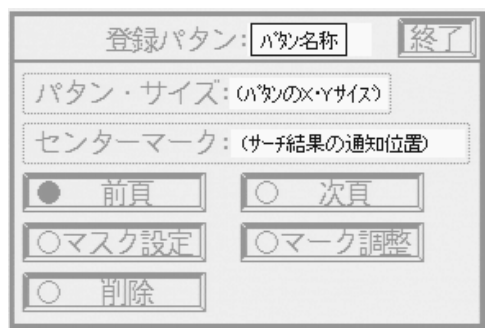
現在登録されている、サーチパタン画像をモニタＴＶに表示します。

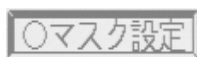
- **サーチパタンの削除**
- **マスク設定（サーチ不感帯の設定）**
- **センターマーク調整（サーチ結果通知位置の調整）**

を行うこともできます。

モニタＴＶには登録されているパタン１個ずつの画像データとともに、下記の項目がグラフィック表示されます。

・パタンの画像範囲・・・・・・・ボックスカーサ

・サーチ結果通知位置・・・・・クロスカーサ

● 前頁 (Backward)
パタンは１画面に１個表示されます。この表示単位を“頁”と呼びます。
“一つ前の頁”のパタンを表示します。

○ 次頁 (Forward)
“次の頁”のパタンを表示します。

○マスク設定 (Mask) **→マスク設定・解除**へ
現在表示されているパタンのマスク（サーチするときの不感帯）設定・解除操作に移ります。

○マーク調整 (Adj. Mark) **→センターマーク調整**へ
現在表示されているパタンのセンターマーク（サーチ結果の通知位置）変更操作に移ります。

○ 削除 (Delete)
現在表示されているパタンを削除します。
選択すると、確認用の「オーバレイパッド」が表示されます。

注　意

パタンデータでマスクが設定されている部分は“黒”（濃度レベル：０）で表示されます。

例

```
#include "f_search.h"

int display_ptn_data()
{
    return( Lib_gs_dsppat(  ));
}
```

留意事項

○表示される画像データの濃度レベル
「サーチ・パタン格納領域」に設定されている、「サーチ・パタン」の画像データは、６ビット（６４階調）ですが、８ビット（２５６階調）に変換して表示されます。
※濃度レベルを４倍して表示します。

Lib_gs_usedel

機　能

ユーザ指定サーチ・パタン削除

形　式

```
#include "f_search.h"
int Lib_gs_usedel( int p_name );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

パラメータで指定される名称の「サーチ・パタン」を削除します。

① ***p_name*** は削除するパタン名称です。
　パタン名称は Lib_gs_defpat, Lib_gs_usepat ライブラリで登録したものを使用します。
　　［注］nullコード(=0x0L)を指定した場合は、全て「サーチ・パタン」を削除します。

戻り値

処理結果

値	意　　　味
0	正常終了しました。
－1	パラメータで指定した名称の「サーチ・パタン」は登録されていません。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

名称“ＡＢＣＤ”なる「サーチ・パタン」を削除します。

```
#include "f_search.h"

int delete_ptn_data()
{
    int name;

    name = 0x44434241;    /*ABCD*/

    return( Lib_gs_usedel( name ));
}
```

留意事項

ありません。

Lib_gs_adjmark

機　能

センター・マーク微調整

形　式

```
#include "f_search.h"
int Lib_gs_adjmark ( int p_name );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

モニタＴＶ表示を見ながらマウスを操作して、登録されている「サーチ・パタン」のセンター・マークＸ／Ｙオフセットを、０．１画素単位で微調整します。
なお、「サーチ・パタン」は２倍、４倍．．．．３２倍に拡大表示する事ができます。

① ***p_name*** はセンター・マークＸ／Ｙオフセットを微調整するパタン名称です。
　パタン名称は Lib_gs_defpat, Lib_gs_usepat ライブラリで登録したものを使用します。

戻り値

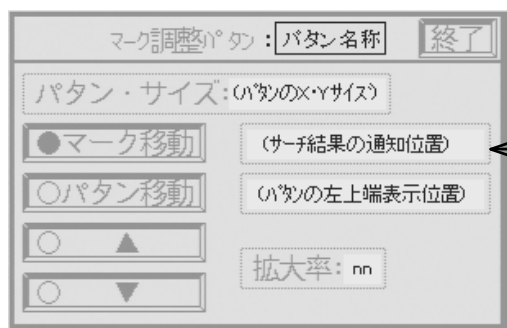
処理結果

値	意　　　味
０	正常終了しました。
１	パネルキー操作のキャンセル指示により、調整処理を取り消します。
－１	パラメータで指定した名称の「サーチ・パタン」は登録されていないので処理ができません。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

設定方法

モニタＴＶには登録されているパタンの画像データとともに、下記の項目がグラフィック表示されます。

・パタンの画像範囲・・・・・・・ボックスカーサ

・サーチ結果通知位置・・・・・クロスカーサ

●マーク移動 (Adj. Mark)

ここを選択すると、トラックボールにより「センターマーク位置」を変更（移動）できます。位置の確定は、左または右ボタンを押します。

○パタン移動 (Move)

ここを選択すると、トラックボールにより「パタンの表示位置」を移動できます。
位置の確定は、左または右ボタンを押します。

(Zoom Rate)

パタンは拡大表示することができます。ｎｎは現在の表示倍率です。
［▲］と［▼］を選択すると表示倍率が変更できます。

例

```
#include "f_search.h"

int set_offset()
{
    int name;
    int rtn_code;

    name = 0x44434241;  /*ABCD*/

    if( 0 == (rtn_code = Lib_gs_defpat( name )) )
    {
        return( Lib_gs_adjmark( name ));
    }
    else
        /*ｴﾗｰ処理*/;
}
```

留意事項

○サーチ・パタン格納領域
　既に、登録済みの「サーチ・パタン」の画像イメージデータが処理の対象となります。

Lib_gs_ptn_get

機　能

パタンデータのアドレス参照

形　式

```
#include "f_search.h"
int Lib_gs_ptn_get( int p_name );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

グレイサーチ・パタンデータのアドレスを参照します。

① ***p_name*** はアドレスを参照する「サーチ・パタン」の名称です。

戻り値

値	定　　数	意　　味
－１	ERROR_RETURN	異常終了しました。
０～	ありません	パタンデータの格納アドレス

例

登録済みサーチ・パタンの縦横サイズを取得します。

```
#include "f_search.h"
#include "f_pinf.h"

static  int             *wbase;        /* ﾜｰﾄﾞ型ﾍﾞｰｽｱﾄﾞﾚｽ */
#define PTNNAME         *(wbase)       /*   +0 ﾊﾟﾀﾝ名称     */
#define P_XSIZE         *(wbase +  1)  /*   +4 xｻｲｽﾞ        */
#define P_YSIZE         *(wbase +  2)  /*   +8 yｻｲｽﾞ        */

void get_ptn_size( int *xsize, int *ysize )
{
     int  name;

     name    = Lib_get_gray_ptn_name(  );
     if( ERROR_RETURN != ( wbase = (int *)Lib_gs_ptn_get( name )))
     {
       *xsize  = P_XSIZE;
       *ysize  = P_YSIZE;
     }
}
```

留意事項

ありません。

Lib_gs_search

機　能

サーチ実行ライブラリ

形　式

```
#include "f_search.h"
int Lib_gs_search ( int dsp_opt, int p_name, int p_cnts, int p_mode,
                    int p_ptype, int p_1score, int p_2score, int *rslt_adrs );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

グレイメモリをサーチして、パラメータで指定される「サーチ・パタン」が存在する座標位置を求めます。
本ライブラリでは、サーチ条件設定(=Lib_gs_scondition)ライブラリと同様な、サーチ実行時の条件を指定できますが、ここで指定する値は今回のサーチに限って有効な条件であり、「サーチ・パタン格納領域」には保存されません。

① ***dsp_opt*** はサーチ位置の表示オプションです。

値	定　　数	意　　味
0	NON_DISPLAY	表示されません。
1	ON_DISPLAY	見つかった位置が表示されます。
９９９	ALL_ON_DISPLAY	候補点と見つかった位置が表示されます。

② ***p_name*** はサーチするパタン名称です。
パタン名称は Lib_gs_defpat, Lib_gs_usepat ライブラリで登録したものを使用します。

③ ***p_cnts*** はサーチ個数です。
サーチする個数を１～５０の数値で指定します。
（サーチ結果格納領域は、この指定値分のエリアを確保する必要があります）

④ ***p_mode*** はサーチ処理の精度です。

値	定　　数	意　　味
1	NORMAL_MODE	通常精度（±２画素（大部分は１画素以内））
2	HIGH_MODE	高精度　（±１画素（大部分は0.5画素以内））
3	SUPER_MODE	超高精度（±0.5画素（大部分は0.1画素以内））

⑤ ***p_ptype*** はサーチ・エリア画像の状態です。範囲は数字１～９の９段階で指定します。
指定方法は、サーチ条件設定(=Lib_gs_scondition)ライブラリを参照してください。

⑥ ***p_1score*** はサーチの第１段階で使用する相関係数です。
範囲は１０００～９９９９の数値で指定します。
指定方法は、サーチ条件設定(=Lib_gs_scondition)ライブラリを参照してください。

⑦ ***p_2score*** はサーチの第２段階で使用する相関係数です。
範囲は１０００～９９９９の数値で指定します。
指定方法は、サーチ条件設定(=Lib_gs_scondition)ライブラリを参照してください。
［注］⑤～⑦のパラメータについて“－１”を指定すると、現在の「サーチ・パタン格納領域」に格納されている各々の設定値が使用されます。
なお、前述のように、本ライブラリの指定では「サーチ・パタン格納領域」の設定値は更新されませんので御留意ください。

⑧ ****rslt_adrs*** はサーチ結果格納バッファアドレスです。
下記の形式で「最終サーチ相関係数値」が指定値以上となる、位置情報を転送。

バイト位置	
Ａ→	相関係数値が1番目のパタンのＸ座標位置（単位：０．１画素）
＋４	相関係数値が1番目のパタンのＹ座標位置（単位：０．１画素）
＋８	相関係数値が1番目のパタンの相関係数値（1000～10000）
＋１２	相関係数値が２番目のパタンのＸ座標位置（単位：０．１画素）
＋１６	相関係数値が２番目のパタンのＹ座標位置（単位：０．１画素）
＋２０	相関係数値が２番目のパタンの相関係数値（1000～10000）
	・　　　　・
	・　　　　・
(N-1)*12	相関係数値がＮ番目のパタンのＸ座標位置（単位：０．１画素）
(N-1)*12+4	相関係数値がＮ番目のパタンのＹ座標位置（単位：０．１画素）
(N-1)*12+8	相関係数値がＮ番目のパタンの相関係数値（1000～10000）

［注］Ｘ・Ｙ座標位置はパタンの左上端に対応する座標ではなく、「センター・マーク」位置に対応する座標です。

戻り値

処理結果

値	意　　味
０～５０	見つけた個数（０の場合は“見つからなかった！！”）
－１	指定した「サーチ・パタン」は登録されていないのでサーチできません。
-999	サーチ・パタン定義エリア指定 (Lib_gs_defadrs)ライブラリが実行されていません。
FVL/LNXをお使いの場合	
－２	サーチパラメータ異常１
－３	サーチパラメータ異常２
-1004	テンポラリバッファの確保に失敗しました。
-1005	内部結果格納バッファが足りません。

またそれぞれのエラーに対する対処法は以下のようになります。

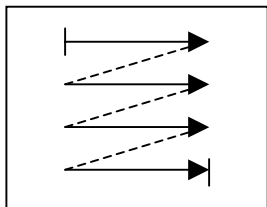
値	対　処　法
－１	「サーチ・パタン」を登録してください。
－２	サーチ個数、精度、複雑度、途中、最終相関値の設定値をチェックしてください。
－３	隣接パタンのX,Y画素数、サーチウィンドウサイズの設定値をチェックしてください。
-999	サーチ･パタン定義エリア指定(Lib_gs_defadrs)ライブラリを実行してください。
-1004	プログラム中で使用していないメモリがあれば解放してください。
-1005	Lib_gs_open()をコールして結果格納バッファを増やしてください。

例

“ＡＢＣＤ”なる「サーチ・パタン」が、３個見つかるまでサーチします。
サーチ方式は、「通常サーチ」、「通常精度」を使用します。
「サーチ・エリアの複雑度」は、既に設定済みの値をそのまま使用します。
「候補点サーチの相関係数下限値」は５５００を使用します。
「最終サーチの相関係数下限値」は７０００を使用します。
サーチの結果として、見つかった座標位置（＝相関係数が７０００以上の点）を、モニタＴＶに表示します。

```
#include "f_search.h"
#include "f_graph.h"

int exec_search()
{
    char ss[50];
    int name;
    int found_cnt;
    int result[9];    /* サーチ個数×３必要です */
    int i;
    double rx, ry;

    name = 0x44434241;   /*ABCD*/

    if( 0 != (found_cnt = Lib_gs_search( NON_DISPLAY, name, 3, NORMAL_MODE,
                                                     1, 5500, 7000, result  )) )
    {
        for( i = 1; i <= found_cnt ; i++ )
        {
            rx = (double)result[(i-1)*3] /10.0;
            ry = (double)result[(i-1)*3+1] /10.0;
            Lib_sprintf( ss, "POINT-%d ( %5.1f, %5.1f ), %04d",
                                             i, rx, ry, result[(i-1)*3+2] );
            Lib_chrdisp( 1, i, ss );
        }
    }
    else
        /*ｴﾗｰ処理*/;
}
```

留意事項

○サーチの方向
サーチの方向は、サーチ範囲の左上端から、右下端に向かって行われます。

○サーチの打切り（終了条件）
サーチ個数見つかった時点で終了するモードと全面サーチ後、相関値の高い順にサーチ個数分返答する２つのモードがあります。
指定方法は Lib_gs_ycondition()ライブラリで行います。

○表示オプションがＯＮの場合は文字用フレームバッファに描画されます。

FVL/LNX

○90X では一回のサーチで検出可能なサーチ個数の上限が 50 個でしたが、FVL/LNX ではデフォルトで 400 個を上限としています。
また一回のサーチで 400 個以上のパタンを検出したい場合は、“Lib_gs_open”（後述参照）を使用してサーチ個数の上限を増やすことが可能です。

○サーチ実行時に指定するパラメータ（複雑度、途中下限値、最終下限値）に－1 を指定できなくなりました。

Lib_gs_xsearch

機　能

連続サーチ実行ライブラリ（条件付き）

形　式

```
#include "f_search.h"
int Lib_gs_xsearch ( int hw_opt, int dsp_opt, int p_name,
                              int p_cnts, int p_mode, int p_ptype,
                                 int p_1score, int p_2score, int *rslt_adrs );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

１回前にサーチを実行した時と同じ状態のグレイメモリをサーチして、パラメータで指定される「サーチ・パタン」が存在する座標位置を求めます。
前述の Lib_gs_search ライブラリでは、まずグレイメモリの１次的な特徴情報を作成後に、指定パタンのサーチ処理を実行しますが、本ライブラリでは前回のサーチ実行時に作成した１次的特徴情報をそのまま流用して、サーチを実行しますので、処理時間が短縮されます。

① ***hw_opt*** は連続処理オプションです。

値	定　　数	意　　味
0x0000	DISAGREE_PTN	サーチ画像は前回と異なる。
0x0001	SAME_IMAGE	サーチ画像は前回と同じです。

② ***dsp_opt*** はサーチ位置の表示オプションです。

値	定　　数	意　　味
0	NON_DISPLAY	表示されません。
1	ON_DISPLAY	見つかった位置が表示されます。
９９９	ALL_ON_DISPLAY	候補点と見つかった位置が表示されます。

③ ***p_name*** はサーチするパタン名称です。
　パタン名称は Lib_gs_defpat，Lib_gs_usepat ライブラリで登録したものを使用します。

④ ***p_cnts*** はサーチ個数です。
　サーチする個数を１～５０の数値で指定します。
　（サーチ結果格納領域は、この指定値分のエリアを確保する必要があります）

⑤ ***p_mode*** はサーチ処理の精度です。

値	定　　数	意　　味
1	NORMAL_MODE	通常精度（±２画素（大部分は１画素以内））
2	HIGH_MODE	高精度（±１画素（大部分は0.5画素以内））
3	SUPER_MODE	超高精度（±0.5画素（大部分は0.1画素以内））

⑥ ***p_ptype*** はサーチ・エリア画像の状態です。範囲は数字１～９の９段階で指示します。
　指定方法は、サーチ条件設定(=Lib_gs_scondition)ライブラリを参照してください。

⑦ ***p_1score*** はサーチの第１段階で使用する相関係数です。
範囲は１０００～９９９９の数値で指定します。
指定方法は、サーチ条件設定(=Lib_gs_scondition)ライブラリを参照してください。

⑧ ***p_2score*** はサーチの第２段階で使用する相関係数です。
範囲は１０００～９９９９の数値で指定します。
指定方法は、サーチ条件設定(=Lib_gs_scondition)ライブラリを参照してください。
［注］⑥～⑧のパラメータについて“－１”を指定すると、現在の「サーチ・パタン格納領域」に格納されている各々の設定値が使用されます。
なお、前述のように、本ライブラリの指定では「サーチ・パタン格納領域」の設定値は更新されませんので御留意ください。

⑨ ****rslt_adrs*** はサーチ結果格納バッファアドレスです。
下記の形式で「最終サーチ相関係数値」が指定値以上となる、位置情報を転送します。

バイト位置

A→	相関係数値が1番目のパタンのＸ座標位置（単位：０．１画素）
＋４	相関係数値が1番目のパタンのＹ座標位置（単位：０．１画素）
＋８	相関係数値が1番目のパタンの相関係数値（1000～10000）
＋１２	相関係数値が２番目のパタンのＸ座標位置（単位：０．１画素）
＋１６	相関係数値が２番目のパタンのＹ座標位置（単位：０．１画素）
＋２０	相関係数値が２番目のパタンの相関係数値（1000～10000）
	・　・
	・　・
(N-1)*12	相関係数値がＮ番目のパタンのＸ座標位置（単位：０．１画素）
(N-1)*12+4	相関係数値がＮ番目のパタンのＹ座標位置（単位：０．１画素）
(N-1)*12+8	相関係数値がＮ番目のパタンの相関係数値（1000～10000）

［注］Ｘ・Ｙ座標位置はパタンの左上端に対応する座標ではなく、「センター・マーク」位置に対応する座標です。

戻り値

処理結果

値	意　味
０～５０	見つけた個数（０の場合は“見つからなかった！！”）
－１	指定した「サーチ・パタン」は登録されていないのでサーチできません。
-990	連続オプション指定のエラーです。
-999	サーチ・パタン定義エリア指定（Lib_gs_defadrs)ライブラリが実行されていません。
FVL/LNXをお使いの場合	
－２	サーチパラメータ異常１
－３	サーチパラメータ異常２
-990	連続オプション指定のエラーです。
-1004	テンポラリバッファの確保に失敗しました。
-1005	内部結果格納バッファが足りません。

またそれぞれのエラーに対する対処法は以下のようになります。

値	対　処　法
－１	「サーチ・パタン」を登録してください。
－２	サーチ個数、精度、複雑度、途中、最終相関値の設定値をチェックしてください。
－３	隣接パタンのX,Y画素数、サーチウィンドウサイズの設定値をチェックしてください。
-990	連続処理オプションの設定値をチェックしてください。
-999	サーチ･パタン定義エリア指定(Lib_gs_defadrs)ライブラリを実行してください。
-1004	プログラム中で使用していないメモリがあれば解放してください。
-1005	Lib_gs_open()をコールして結果格納バッファを増やしてください。

例

“ＡＢＣＤ”の「サーチ・パタン」をサーチ後に、同一条件で“ＥＦＧＨ”の「サーチ・パタン」をサーチします。

```
#include "f_search.h"

int exec_search()
{
    int name1;
    int name2;
    int found_cnt1;
    int found_cnt2;
    int result1[9];
    int result2[9];

    name1 = 0x44434241;/*ABCD*/
    name2 = 0x48474645;/*EFGH*/

    found_cnt1 = Lib_gs_search( NON_DISPLAY, name1, 3, NORMAL_MODE, 1,
                                                    5500, 7000, result1  );
    found_cnt2 = Lib_gs_xsearch( SAME_IMAGE, NON_DISPLAY,name2, 3,
                                      NORMAL_MODE, 1, 5500, 7000, result2 );
}
```

留意事項

○グレイメモリ
　連続オプションで‘同一シーン’が指定される場合は、前回のサーチ実行時と同じ８ビット濃淡画像データが格納されていなければいけません。

FVL/LNX

○90X では一回のサーチで検出可能なサーチ個数の上限が 50 個でしたが、FVL/LNX ではデフォルトで 400 個を上限としています。
また一回のサーチで 400 個以上のパタンを検出したい場合は、“Lib_gs_open”(後述参照)を使用してサーチ個数の上限を増やすことが可能です。

○サーチ実行時に指定するパラメータ(複雑度、途中下限値、最終下限値)に－1 を指定できなくなりました。

Lib_gs_ysearch

機　能　　回転サーチ実行ライブラリ（条件付き）

形　式

```
#include "f_search.h"
int Lib_gs_ysearch( int rot_opt, int continue_opt, int dsp_opt, int p_name,
                    int p_cnts, int p_mode, int p_ptype, int p_1score,
                    int p_2score, int *rslt_adrs );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

グレイメモリをサーチして、パラメータで指定される回転した状態の「サーチ・パタン」が存在する座標位置を求めます。
また、Lib_gs_xsearch ライブラリと同様に、サーチ時間を短縮する‘連続オプション’を指定することもできます。

① ***rot_opt*** は回転オプションです。
「サーチ・パタン」の中心を原点とした回転角度を０．１度単位で指定します。
指定可能範囲：－３５５９～＋３５５９（時計方向：正、反時計方向：負）

② ***continue_opt*** は連続処理オプションです。

③ ***dsp_opt*** はサーチ位置の表示オプションです。

④ ***p_name*** はサーチするパタン名称です。
パタン名称は Lib_gs_defpat, Lib_gs_usepat ライブラリで登録したものを使用します。

⑤ ***p_cnts*** はサーチ個数です。
サーチする個数を１～５０の数値で指定します。
（サーチ結果格納領域は、この指定値分のエリアを確保する必要があります）

⑥ ***p_mode*** はサーチ処理の精度です。

⑦ ***p_ptype*** はサーチ・エリア画像の状態です。範囲は数字１～９の９段階で指示します。
指定方法は、サーチ条件設定(=Lib_gs_scondition)ライブラリを参照してください。

⑧ ***p_1score*** はサーチの第１段階で使用する相関係数です。
範囲は１０００～９９９９の数値で指定します。
指定方法は、サーチ条件設定(=Lib_gs_scondition)ライブラリを参照してください。

⑨ ***p_2score*** はサーチの第２段階で使用する相関係数です。
範囲は１０００～９９９９の数値で指定します。
指定方法は、サーチ条件設定(=Lib_gs_scondition)ライブラリを参照してください。
［注］⑦～⑨のパラメータについて“－１”を指定すると、現在の「サーチ・パタン格納領域」に格納されている各々の設定値が使用されます。
なお、前述のように、本ライブラリの指定では「サーチ・パタン格納領域」の設定値は更新されませんので御留意ください。

⑩ ***rslt_adr** はサーチ結果格納バッファアドレスです。
下記の形式で「最終サーチ相関係数値」が指定値以上となる、位置情報を転送します。

バイト位置

バイト位置	内容
A→	相関係数値が1番目のパタンのＸ座標位置（単位：０．１画素）
＋４	相関係数値が1番目のパタンのＹ座標位置（単位：０．１画素）
＋８	相関係数値が1番目のパタンの相関係数値（1000～10000）
＋１２	相関係数値が２番目のパタンのＸ座標位置（単位：０．１画素）
＋１６	相関係数値が２番目のパタンのＹ座標位置（単位：０．１画素）
＋２０	相関係数値が２番目のパタンの相関係数値（1000～10000）
	・　　　・
	・　　　・
(N-1)*12	相関係数値がＮ番目のパタンのＸ座標位置（単位：０．１画素）
(N-1)*12+4	相関係数値がＮ番目のパタンのＹ座標位置（単位：０．１画素）
(N-1)*12+8	相関係数値がＮ番目のパタンの相関係数値（1000～10000）

［注］Ｘ・Ｙ座標位置はパタンの左上端に対応する座標ではなく、「センター・マーク」位置に対応する座標です。

戻り値

処理結果

値	意　味
０～５０	見つけた個数（０の場合は“見つからなかった！！”）
－１	指定した「サーチ・パタン」は登録されていないのでサーチできません。
-990	連続オプション指定のエラーです。
-999	サーチ・パタン定義エリア指定（Lib_gs_defadrs）ライブラリが実行されていません。
FVL/LNXをお使いの場合	
－２	サーチパラメータ異常１
－３	サーチパラメータ異常２
-990	連続オプション指定のエラーです。
-991	回転処理オプション指定のエラーです。
-1003	回転サーチパタンバッファの確保に失敗しました。
-1004	テンポラリバッファの確保に失敗しました。
-1005	内部結果格納バッファが足りません。

またそれぞれのエラーに対する対処法は以下のようになります。

値	対　処　法
－１	「サーチ・パタン」を登録してください。
－２	サーチ個数、精度、複雑度、途中、最終相関値の設定値をチェックしてください。
－３	隣接パタンのX,Y画素数、サーチウィンドウサイズの設定値をチェックしてください。
-990	連続処理オプションの設定値をチェックしてください。
-991	回転処理オプションの設定値をチェックしてください。
-999	サーチ・パタン定義エリア指定（Lib_gs_defadrs）ライブラリを実行してください。
-1003	プログラム中で使用していないメモリがあれば解放してください。
-1004	プログラム中で使用していないメモリがあれば解放してください。
-1005	Lib_gs_open()をコールして結果格納バッファを増やしてください。

例

“ＡＢＣＤ”の「サーチ・パタン」をサーチ後に、同一条件で“ＥＦＧＨ”の「サーチ・パタン」をサーチします。

```
#include "f_search.h"

int exec_search()
{
    int name1;
    int name2;
    int found_cnt1;
    int found_cnt2;
    int result1[9];
    int result2[9];

    name1 = 0x44434241;/*ABCD*/
    name2 = 0x48474645;/*EFGH*/

    found_cnt1 = Lib_gs_search( NON_DISPLAY, name1, 3, NORMAL_MODE, 1,
                                                        5500, 7000, result1  );
    found_cnt2 = Lib_gs_ysearch( 1800, SAME_IMAGE, NON_DISPLAY, name2, 3,
                                             NORMAL_MODE, 1, 5500, 7000, result2 );
}
```

留意事項

○グレイメモリ
　連続オプションで‘同一シーン’が指定される場合は、前回のサーチ実行時と同じ８ビット濃淡画像データが格納されていなければいけません。

FVL/LNX

○90X では一回のサーチで検出可能なサーチ個数の上限が 50 個でしたが、FVL/LNX ではデフォルトで 400 個を上限としています。
　また一回のサーチで 400 個以上のパタンを検出したい場合は、“Lib_gs_open”(後述参照)を使用してサーチ個数の上限を増やすことが可能です。

○サーチ実行時に指定するパラメータ(複雑度、途中下限値、最終下限値)に－1 を指定できなくなりました。

Lib_gs_psearch

機　能　　サーチ実行ライブラリⅡ

形　式

```
#include "f_search.h"
int Lib_gs_psearch ( int dsp_opt, int p_no, int *rslt_adrs );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

システム・パラメータのサーチパラメータをサーチ条件として、「サーチ・パタン」が存在する座標位置を求めます。

① ***dsp_opt*** はサーチ位置の表示オプションです。

値	定　数	意　味
0	NON_DISPLAY	表示されません。
1	ON_DISPLAY	見つかった位置が表示されます。
９９９	ALL_ON_DISPLAY	候補点と見つかった位置が表示されます。

② ***p_no*** はサーチ条件となるグレイサーチ処理パラメータ（アクター番号）番号です。（０～１２７）
パタン名称は Lib_gs_defpat, Lib_gs_usepat ライブラリで登録したものを使用します。

③ ****rslt_adrs*** はサーチ結果格納バッファアドレスです。
下記の形式で「最終サーチ相関係数値」が指定値以上となる、位置情報を転送します。

バイト位置	
A→	相関係数値が1番目のパタンのＸ座標位置（単位：０．１画素）
＋４	相関係数値が1番目のパタンのＹ座標位置（単位：０．１画素）
＋８	相関係数値が1番目のパタンの相関係数値（1000～10000）
＋１２	相関係数値が２番目のパタンのＸ座標位置（単位：０．１画素）
＋１６	相関係数値が２番目のパタンのＹ座標位置（単位：０．１画素）
＋２０	相関係数値が２番目のパタンの相関係数値（1000～10000）
	・　　　・
	・　　　・
(N-1)*12	相関係数値がＮ番目のパタンのＸ座標位置（単位：０．１画素）
(N-1)*12+4	相関係数値がＮ番目のパタンのＹ座標位置（単位：０．１画素）
(N-1)*12+8	相関係数値がＮ番目のパタンの相関係数値（1000～10000）

［注］Ｘ・Ｙ座標位置はパタンの左上端に対応する座標ではなく、「センター・マーク」位置に対応する座標です。

戻り値

処理結果

値	意　　　　　　　味
０～５０	見つけた個数（０の場合は“見つからなかった！！”）
－１	指定した「サーチ・パタン」は登録されていないのでサーチできません。
-999	サーチ・パタン定義エリア指定 (Lib_gs_defadrs)ライブラリが実行されていません。

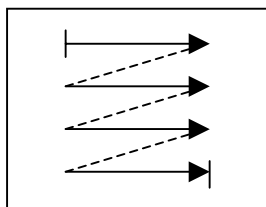
例

グレイサーチ処理パラメータの０番を使用し、サーチします。

```
#include "f_search.h"
#include "f_graph.h"

#define MAX_NO_GRAY_SEARCH    50

int exec_search()
{
    char ss[50];
    int found_cnt;
    int result[MAX_NO_GRAY_SEARCH * 3];    /* サーチ個数×３必要です */
    int i;
    double rx, ry;

    if( 0 != ( found_cnt = Lib_gs_psearch( NON_DISPLAY, 0, result ) ) )
    {
        for( i = 1; i <= found_cnt ; i++ )
        {
            rx = (double)result[(i-1)*3] /10.0;
            rx = (double)result[(i-1)*3+1] /10.0;
            Lib_sprintf( ss, "POINT-%d ( %5.1f, %5.1f ), %04d",
                                      i, rx, ry, result[(i-1)*3+2] );
            Lib_chrdisp( 1, i, ss );
        }
    }
    else
        /*ｴﾗｰ処理*/;
}
```

留意事項

○サーチの方向

サーチの方向は、サーチ範囲の左上端から、右下端に向かって行われます。

○サーチの打切り（終了条件）

サーチ個数見つかった時点で終了するモードと全面サーチ後、相関値の高い順にサーチ個数分返答する２つのモードがあります。

指定方法は Lib_gs_yconditionライブラリで行います。

○表示オプションがＯＮの場合は文字用フレームバッファに描画されます。

Lib_gs_point_search

機　能

詳細サーチ実行ライブラリ

形　式

```
#include "f_search.h"
int Lib_gs_point_search( int p_name, int w_xs, int w_ys,
                                        int w_xe, int w_ye, int *rslt_adrs );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

指定範囲のグレイメモリをサーチして、パラメータで指定される「サーチ・パタン」が存在する一番相関値が高い座標を求めます。

① ***p_name*** はサーチするパタン名称です。
　パタン名称は Lib_gs_defpat，Lib_gs_usepat ライブラリで登録したものを使用します。

② ***w_xs*** はサーチ・ウインドウの左上端Ｘ座標です。

③ ***w_ys*** はサーチ・ウインドウの左上端Ｙ座標です。

④ ***w_xe*** はサーチ・ウインドウの右下端Ｘ座標です。

⑤ ***w_ye*** はサーチ・ウインドウの右下端Ｙ座標です。

⑥ ****rslt_adrs*** はサーチ結果格納バッファアドレスです。
　下の形式で位置情報を転送します。

バイト位置

Ａ→	パタンのＸ座標位置（単位：0.１画素）
＋４	パタンのＹ座標位置（単位：0.１画素）
＋８	パタンの相関係数値（０～１００００）

［注］Ｘ・Ｙ座標位置はパタンの左上端に対応する座標ではなく、「センター・マーク」に対応する座標です。

戻り値

処理結果

値	意　　味
１	パタンが見つかり正常終了しました。
０	パタンは見つかりませんでした。
－１	パラメータ指定の名称の「サーチ・パタン」は登録されていないので処理できません。
－２	パラメータ・エラー（処理範囲が不正）です。
－３	ワークメモリが取得できません。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

”ＡＢＣＤ”なる「サーチ・パタン」を左上座標（１００，１５０）右下座標（２００，２５０）の指定「サーチ・ウインドウ」から１個探します。

```
#include "f_search.h"

int test_point_match()
{
    int name;
    int result[3];

    name = 0x44434241;   /* ABCD */
    return ( Lib_gs_point_search( name, 100, 150, 200, 250, result ) );
}
```

留意事項

○本ライブラリはサーチ精度を高めるため画像圧縮処理等を行っていません。
　そのためサーチ・ウインドウを大きくとりますと処理時間がかなり遅くなります。

○処理時間の目安
　Lib_gs_pcorr × ｛ ｘ（ サーチウインドＸサイズー登録パタンＸサイズ ）
　　　　　　　　　　　ｙ（ サーチウインドＹサイズー登録パタンＹサイズ)＋α ｝

Lib_gs_gfreeze

機　能

１次的特徴情報作成

形　式

```
#include "f_search.h"
int Lib_gs_gfreeze( int flag );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

連続サーチ実行(=Lib_gs_xsearch())ライブラリで使用するためのグレイメモリの１次的な特徴情報を作成します。

① ***flag*** は１次特徴情報タイプ（サーチモード）です。

値	定　　数	意　　味
0	NORMAL_PATTERN	サーチ実行(=GS_SEARCH)ライブラリ実行時と同じ、１次的な特徴情報を作成します。
1	BLACK_LINE_PATTERN	黒い線状パタンのサーチを実施する。
2	WHITE_LINE_PATTERN	白い線状パタンのサーチを実施する。
3	UNSTABLE_PATTERN	標準タイプではサーチ処理結果が不安定(見つからない事がある)な場合にご使用ください。

戻り値

処理結果

値	意　　味
0	正常終了しました。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

Lib_gs_xsearch() と組み合わせて、Lib_gs_search() と同等の処理を実行します。

```
#include "f_search.h"

int exec_search()
{
    int name;
    int found_cnt;
    int result[9];
    int out_inf[8];

    name = 0x44434241;/*ABCD*/

    if ( NORMAL_RETURN == Lib_gs_infpat( name, out_inf ) )
    {
        Lib_gs_gfreeze( out_inf[6] );
        if( 0 != ( found_cnt = Lib_gs_xsearch( SAME_IMAGE,
                    NON_DISPLAY, name, 3, NORMAL_MODE, 1, 5500, 7000, result )))
        {
            /*結果表示*/;
        }
        else
            /*ｴﾗｰ処理*/;
     }
     else
         /*ｴﾗｰ処理*/;
}
```

留意事項

○サーチパタン登録時のサーチモードは Lib_gs_infpat にて取得できます。

Lib_gs_pcorr

機　能

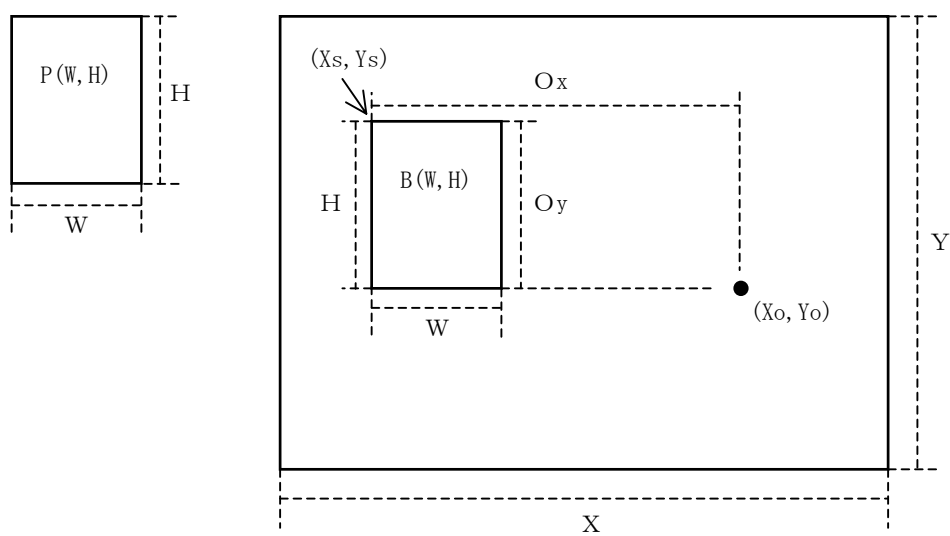
相関値計算ライブラリ（１点マッチング）

形　式

```
#include ″f_search.h″
int Lib_gs_pcorr ( int p_mode, int p_name, int p_xadrs,
                              int p_yadrs, int *rslt_adrs );
```

９０１	９０２	９０３	９０４	高分解能
◯	◯	◯	◯	◯

解　説

「サーチ・パタン」と、グレイメモリの部分画像との相関係数値を求めます。

但し、（Xs，Ys）は「サーチ・パタン」を矩形とした場合の左上端位置。
（Xo，Yo）は「センター・マーク」位置。（サーチ実行結果として返答される位置）
（Ox，Oy）は、それぞれセンターマークX，Yオフセットとする。

① ***p_mode*** は相関位置の指定座標モードです。

値	定　数	意　味
0	BASIC_POS	Xs，Ysを指定座標とします。
1	CENTER_POS	Xo，Yoを指定座標とします。

② ***p_name*** は相関係数を求めるパタン名称です。
パタン名称は Lib_gs_defpat,Lib_gs_usepat ライブラリで登録したものを使用します。

③ ***p_xadrs*** はX座標位置です。
- 「サーチ・パタン」を矩形とした場合の左上端X座標（Xs）
  または、
- 「センターマーク」のX座標（Xo）

④ ***p_yadrs*** はＹ座標位置です。
・「サーチ・パタン」を矩形とした場合の左上端Ｙ座標（Ｙs）
　または、
・「センターマーク」のＹ座標（Ｙo）
［注］③④によってフレームメモリの部分画像を指定しますが、「センターマーク」の場合は単位を０．１画素で指定してください。

⑤ ****rslt_adrs*** は相関係数値です。結果は０から１００００のワード形式にて返送します。

戻り値

処理結果

値	意　　　　味
０	正常終了しました。
－１	パラメータで指定した名称の「サーチ・パタン」は登録されていないので処理ができません。
－２	入力パラメータで指定したＸ・Ｙ座標に該当する部分画像は、メモリ範囲外です。
-999	サーチ･パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

“ＡＢＣＤ”なる「サーチ・パタン」と、フレームメモリの部分画像の相関係数値を計算します。
部分画像の位置は、ｘ座標＝１００、ｙ座標＝１５０を左上端とした矩形領域です。

```
#include "f_search.h"

int set_pmatch()
{
    int name;
    int rtn_code;
    int corr_val;

    name = 0x44434241;   /*ABCD*/
    return ( Lib_gs_pcorr( BASIC_POS, name, 100, 150, &corr_val ));
}
```

留意事項

○グレイメモリ
　現在グレイメモリに格納されている８ビット濃淡画像データが処理の対象となります。

○「サーチ・パタン」の画像データ及び相関係数計算用データ
　現在「サーチ・パタン格納領域」に設定されている
　・パラメータで指定した「サーチ・パタン」の画像データ
　・パラメータで指定した「サーチ・パタン」の相関係数計算用データ
　を使用して相関係数を求めます。

Lib_gs_window

機　能

サーチ・エリア指定ライブラリ

形　式

```
#include "f_search.h"
int Lib_gs_window( int dsp_opt, int w_xs, int w_ys, int w_xe, int w_ye );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

サーチを実行する「サーチ・ウィンドウ」を指定するもので、設定内容は１度設定すると再度、本ライブラリを実行するまで保存されます。
本ライブラリにて許容される「サーチ・ウィンドウ」は、矩形に限定されます。
「サーチ・ウィンドウ」は下記の働きをします。

サーチ処理の範囲を限定
サーチ実行(=Lib_gs_search)ライブラリでは「サーチ・ウィンドウ」にて限定される範囲の矩形領域のみが処理対象となります。
（「サーチ・ウィンドウ」の範囲が狭いほど、当然サーチに要する時間は短くなります)

① ***dsp_opt*** は「サーチ・ウィンドウ」の表示オプションです。

値	定　　数	意　　　味
0	NON_DISPLAY	白枠を表示しません。
1	ON_DISPLAY	白枠を表示します。

② ***w_xs*** はサーチ・ウィンドウの左上端Ｘ座標です。
範囲は ０～Lib_get_fx_size( )－８ の数値で指定します。
［注］“－１”を指定すると以下のパラメータは全てダミーとなります。
これは、現在の設定中の「サーチ・ウィンドウ」範囲は変更せずに、単に、ウィンドウ形状のみ表示したい場合などに用いられます。

③ ***w_ys*** はサーチ・ウィンドウの左上端Ｙ座標です。
範囲は ０～Lib_get_fy_size( )－８ の数値で指定します。

④ ***w_xe*** はサーチ・ウィンドウの右下端Ｘ座標です。
範囲は ７～Lib_get_fx_size( )－１ の数値で指定します。

⑤ ***w_ye*** はサーチ・ウィンドウの右下端Ｙ座標です。
範囲は ７～Lib_get_fy_size( )－１ の数値で指定します。

戻り値

処理結果

値	意　　　味
０	正常終了しました。
－１	指定された座標値が範囲外です。
－２	指定したサーチ範囲の幅または高さが８画素未満です。
-999	サーチ･パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

ステージパラメータの設定範囲を参照し、実行する。

```
#include "f_search.h"
#include "f_pinf.h"

void window()
{
    int xs, ys, xe, ye;

    Lib_get_stage_window( &xs, &ys, &xe, &ye );
    Lib_gs_window( ON_DISPLAY, xs, ys, xe, ye );
}
```

留意事項

○サーチするパタンのサイズとの関連
「サーチ・ウィンドウ」の範囲を小さくすると、サーチ実行(=Lib_gs_search)ライブラリの処理速度が向上しますが、サーチするパタンのサイズより「サーチ・ウィンドウ」の範囲が小さい場合は、サーチが不成功になりますのでご注意ください。
また、「サーチ・ウィンドウ」に接するパタンは検出されません。

○サーチ・パタン格納領域
サーチ・パタン格納領域の「サーチ・ウィンドウ」が新たなデータにより更新されます。

Lib_gs_scondition

機　能

サーチ条件設定ライブラリ

形　式

```
#include "f_search.h"
int Lib_gs_scondition ( int p_stype, int p_1thresh, int p_2thresh );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

サーチを実行する時の各種処理条件を規定するもので、ここで指定する値は
・サーチ速度
・サーチの確実性（期待したパタンと異なる物をみつけてしまう）
に影響をおよぼします。
なお、本ライブラリで指定した処理条件は「サーチ・パタン格納領域」に保存されます。
サーチ処理は下記の２段階で行われます。
・第１段階‥‥‥ 「候補点サーチ」と呼びます。
近似的な相関計算を行い「サーチ・パタン」と似ている図形を探し出す。（ここで探しだした位置を“候補点”と呼びます。）
・第２段階‥‥‥ 「最終サーチ」と呼びます。
“候補点”について相関計算を行い、最終的に「見つけた」と判断する。

① ***p_stype*** は「候補点サーチ」で行う、近似的な相関計算に使用するデータ量の度合いを１～９で指定します。
‘１’が最もデータ量が少なくなりサーチ時間が短縮されます。
（現バージョンでは‘１’‘５’‘９’の３段階が有効です。）
一般的には、画像の状態を下記のような要素から判断して指定します。
◎ 外乱光・ノイズ等も無く安定した画像かどうか。
◎ 背景は濃度レベルが一定な画像か。
◎「サーチ・パタン」と背景のコントラストがあるか。
（１が最も良い状態を意味します…人間が見て、非常に簡単に探せると思われる状態）
通常は“５”（＝初期設定される値）を指定しますが、小さい値を指定するとサーチ速度の向上が期待できる反面、予期せぬパタンを探してしまう危険性もあります。

② ***p_1thresh*** は「候補点サーチ」で使用する相関係数値のしきい値を 1000～9999 で指定します。本設定値によりサーチ速度がかなり変動します。
低くする‥‥‥‥ 「候補点」が多数検出され、「最終サーチ」の計算量が増えるので、サーチ時間がかかるようになります。
但し、確実に見つけることができます。

高くする‥‥‥‥ 「候補点」の数が少なくなり、「最終サーチ」の計算量が減るので、サーチ時間が短くなります。
但し、見つけたいパタンが、見つからなくなる危険性があります。

③ ***p_2thresh*** は「最終サーチ」で使用する相関係数値のしきい値を1000～9999で指定します。最終的に“見つけた”と判断するしきい値です。
サーチすべき画像の最悪の状態を考慮して指定します。
つまり
・ノイズ、フォーカスずれによる画像の劣化
・パタンのカケ、大きさの変化、回転により、相関係数値が低くなることを想定して指定してください。（良品・不良品・欠品の検査をする場合は、良品の限度値として使用する事もできます。）

※通常は、「最終サーチの相関係数下限値」＞「候補点サーチの相関係数下限値」を、設定しますが、「サーチ・パタン」以外にほとんどパタン（図形）が無い時などは、「候補点サーチの相関係数下限値」のほうを大きく設定することにより、速度を低下させずに安定したサーチを実行する事もできます。
［注］パラメータ値として“－１”を指定すると、そのパラメータ値については、前回指定値をそのまま流用する、と言った意味となります。

戻り値

処理結果

値	意　　味
0	正常終了しました。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

グレイ処理パラメータを参照し、実行する。

```
#include "f_search.h"
#include "f_pinf.h"

void condition()
{
    int complex, mid_lower, last_lower;

    complex    = Lib_get_gray_complex( );
    mid_lower  = Lib_get_gray_mid_lower( );
    last_lower = Lib_get_gray_last_lower( );

    Lib_gs_scondition( complex, mid_lower, last_lower );
}
```

留意事項

○パラメータ指定値が範囲外の場合
範囲外のパラメータ値が指定された場合は、それぞれの最大値または最小値がセットされます。
◎「サーチ・エリアの複雑度」･･････････ 最大値＝９，　最小値＝１
◎「候補点サーチの相関係数下限値」･････ 最大値＝９９９９，　最小値＝１０００
◎「最終サーチの相関係数下限値」･･･････ 最大値＝９９９９，　最小値＝１０００

○サーチ・パタン格納領域
サーチ・パタン格納領域が指定したパラメータ値で更新されます。
（“－１”が指定された項目以外）

○精度について
分解能を下げる事により、処理速度を速くする事を実現しています。
“通常精度”が最も速く、“高精度”、“超高精度”となるに従って遅くなります。

Lib_gs_xcondition

機　能

サーチ条件設定ライブラリⅡ

形　式

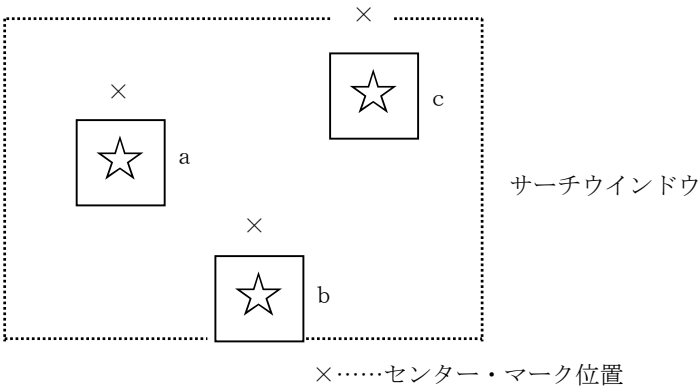
```
#include "f_search.h"
int Lib_gs_xcondition ( int edge_detect, int x_step,
                        int y_step, int line_width, int point_width );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

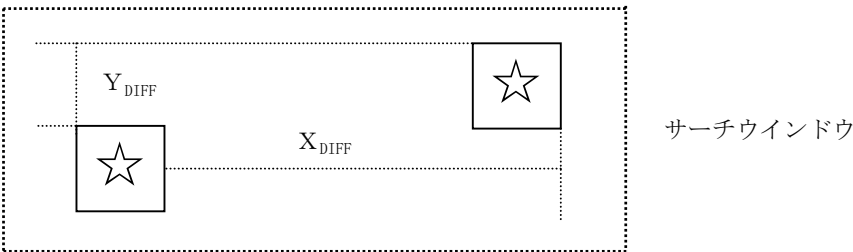
解　説

サーチ実行時に関する、下記の条件を設定するものです。
- ・ウィンドウに接するパタンの抽出有無
- ・複数個のパタン検出時の抽出条件（検出位置によるソーティング）

なお、本ライブラリで設定した処理条件は「サーチ・パタン格納領域」に保存されます。
また、サーチ・パタン定義(Lib_gs_defadrs)ライブラリ実行時（但し、初期化モードの場合）、システム既定値が設定されます。

① ***edge_detect*** はウィンドウ接触パタン検出有無オプションです。
　ゼロ ‥‥‥‥‥ サーチ・ウィンドウに接するパタンを除去します。（システム既定値）
　ノンゼロ ‥‥‥ サーチ・ウィンドウに接するパタンを抽出します。
　［注］チェックされるのは、パタンの座標位置で、センターマークの座標位置ではありません。

ａ，ｃ ‥‥‥‥‥ ウィンドウに接しないパタン。
ｂ ‥‥‥‥‥‥‥ ウィンドウに接するパタン。

② ***x_step*** は隣接パタンのＸ方向画素数（０～５１２）
　複数個のパタンを検出した場合、その座標位置が本パラメータ以内の場合は相関値の大きな方のパタンを採用します。
　［注］ゼロを指定すると、サーチするパタンサイズの１／２の値が自動的に使用されます。（システム既定値です。）

③ ***y_step*** は隣接パタンのＹ方向画素数（０～４８０）です。

XDIFF ＜ 隣接パタンのＸ方向画素数　かつ
YDIFF ＜ 隣接パタンのＹ方向画素数
の場合はサーチ結果は１個。
その他の場合はサーチ結果は２個となります。

④ ***line_width*** はラインサーチの幅です。（ダミー）

⑤ ***point_width*** はポイントサーチの幅です。（ダミー）

戻り値

処理結果

値	意　　　味
0	正常終了しました。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

ウィンドウに接するパタンを抽出します。

```
#include "f_search.h"
#include "f_pinf.h"
#define  DUMMY  0

void xcondition()
{
    Lib_gs_xcondition( 0xff, DUMMY, DUMMY, DUMMY, DUMMY );
}
```

留意事項

○システム既定値
　サーチ・パタン定義(Lib_gs_defadrs)ライブラリ実行時（但し、初期化モードの場合）、
　　・ウィンドウ接触パタン検出有無･･･････　接するパタンを除去する。
　　・隣接パタンＸ方向画素数･････････････　サーチするパタンサイズの１／２
　　・隣接パタンＹ方向画素数･････････････　サーチするパタンサイズの１／２
　のシステム既定値が設定されます。

○ Lib_gs_window をあらかじめ実行してください。

Lib_gs_ycondition

機　能

サーチ条件設定ライブラリⅢ

形　式

```
#include "f_search.h"
int Lib_gs_ycondition ( int p0, int p1, int p2, int p3, int p4 );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

サーチ実行時に関する、下記の条件を設定するものです。
・サーチ終了方式

なお、本ライブラリで設定した処理条件は「サーチ・パタン格納領域」に保存されます。
また、サーチ・パタン定義（Lib_gs_defadrs)ライブラリ実行時（但し、初期化モードの場合）、システム既定値が設定されます。

① ***p0*** は、ダミーデータです。（必ず０を指定してください。）

② ***p1*** は、サーチ終了方式です。

値	定　　数	意　　味
1	ありません	サーチ個数が指定した個数に達した時点でサーチを打ち切り、相関値の高い順に返答します。
上記以外	ありません	全面サーチ後、相関値の高い順に返答します。

③ ***p2*** は、サーチ結果出力方式です。

値	定　　数	意　　味
0	ありません	サーチ結果(X,Y座標)を10倍値で出力します。
１００	ありません	サーチ結果(X,Y座標)を100倍値で出力します。

④ ***p3*** は、ダミーデータです。（必ず０を指定してください。）

⑤ ***p4*** は、ダミーデータです。（必ず０を指定してください。）

戻り値

値	意　　味
0	正常終了しました。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

旧バージョンとの互換を維持

```
#include "f_search.h"
#include "f_gui.h"
#define  PTN_NAME     0x44434241
#define  PTN_NO       1
#define  GSNMB        10
#define  DUMMY        0
#define  OLD_VERSION  1

static int rslt_buf[GSNMB][3];

void main()
{
    static char string[30];
    int i, rtn_code;
    int x, y, a, amax;

    Lib_init_cursor();
    Lib_gs_open_data_file( INITIAL_PTN_AREA );
    Lib_gs_ycondition ( DUMMY, OLD_VERSION, DUMMY, DUMMY, DUMMY );
    Lib_gs_defpat( PTN_NAME );

    if( CURSOR_EXECUTE == Lib_display_keyinput( 10, 450, "ｻｰﾁ実行" ))
    {
          if( 0 < (rtn_code = Lib_gs_search( ON_DISPLAY, PTN_NAME,
                      PTN_NO, NORMAL_MODE, -1, -1, -1,(int *)rslt_buf )) )
          {
              for (i = 0, amax = 0; i < rtn_code; i++)
                                              /*相関値の一番高いﾊﾟﾀﾝを抽出*/
              {
                   a = rslt_buf[i][2];
                   if (amax < a)
                   {
                        amax = a;
                        x = rslt_buf[i][0];
                        y = rslt_buf[i][1];
                   }
              }
              Lib_printf( string, "X位置:%4d", x    );
              Lib_chrdisp( 2, 5, string );
              Lib_printf( string, "Y位置:%4d", y    );
              Lib_chrdisp( 2, 6, string );
              Lib_printf( string, "相関値:%4d", amax );
              Lib_chrdisp( 2, 7, string );
          }
    }
    Lib_gs_save_data_file(  );
    Lib_gs_close_data_file(  );
}
```

留意事項

○システム既定値

・サーチ終了方式‥‥‥‥‥ 全面サーチ後、相関値の高い順に返答します。

・サーチ結果出力方式‥‥‥ サーチ結果(X,Y座標)を10倍値で出力します。

○サーチ結果を100倍値にした場合、±0.05画素程度の精度が得られます。

Lib_gs_smode

機　能

特殊サーチ制御

形　式

```
#include "f_search.h"
void Lib_gs_smode( int  mode );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

サーチ・パタンが細い線状であるとか、サーチ処理結果が不安定（相関値が一定せず見つからない事がある）な場合に、本ライブラリを実施後、パタンを再登録します。

① ***mode*** は、特殊サーチモードです。

値	定　　数	意　　味
０	NORMAL_PATTERN	通常のパタンの登録・サーチを実施する。
１	BLACK_LINE_PATTERN	黒い線状パタンのサーチを実施する。
２	WHITE_LINE_PATTERN	白い線状パタンのサーチを実施する。
３	UNSTABLE_PATTERN	通常では、サーチ処理結果が不安定なパタンの登録

戻り値

なし

例

黒い線状のものを登録後、サーチします。

```
#include "f_search.h"
#include "f_gui.h"
#define  PTN_NAME     0x44434241
#define  PTN_NO       1

void main()
{
    static int rslt[2][3];
    static char string[30];

    Lib_init_cursor();
    Lib_gs_open_data_file( CONTINUE_PTN_AREA );
    Lib_gs_smode( BLACK_LINE_PATTERN );
    Lib_gs_defpat( PTN_NAME );

    if( CURSOR_EXECUTE == Lib_display_keyinput( 10, 450, "ｻｰﾁ実行" ))
    {
        if( 0 < Lib_gs_search( ON_DISPLAY, PTN_NAME, PTN_NO,
                               NORMAL_MODE, -1, -1, -1, (int *)rslt ))
        {
            Lib_printf( string, "X位置:%4d", rslt[0][0] );
            Lib_chrdisp( 2, 5, string );
            Lib_printf( string, "Y位置:%4d", rslt[0][1] );
            Lib_chrdisp( 2, 6, string );
            Lib_printf( string, "相関値:%4d", rslt[0][2] );
            Lib_chrdisp( 2, 7, string );
        }
    }
    Lib_gs_save_data_file(  );
    Lib_gs_close_data_file(  );
}
```

留意事項

○サーチ実行時は、サーチパタンの登録時点で記憶したサーチモードで自動的にサーチされますので、サーチ実行ライブラリ(Lib_gs_search等)の直前で、当ライブラリをあらためてコールする必要はありません。

Lib_gs_infpat

機　能

サーチ・パタン情報ＧＥＴライブラリ

形　式

```
#include "f_search.h"
int Lib_gs_infpat( int p_name, int out_inf[8] );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

登録済み「サーチ・パタン」の画像サイズ、センター・マークＸ／Ｙオフセット位置を、取り出します。

① ***p_name*** は情報を取り出すパタン名称です。
パタン名称は Lib_gs_defpat, Lib_gs_usepat ライブラリで登録したものを使用します。

② ***out_inf[8*** はパタン情報格納アドレス（ポインタ）です。
out_inf[0] ・・・・・・・・・・・・・・ パタン横サイズ
out_inf[1] ・・・・・・・・・・・・・・ パタン縦サイズ
out_inf[2] ・・・・・・・・・・・・・・ センターマークｘオフセット値（単位：０．１画素）
out_inf[3] ・・・・・・・・・・・・・・ センターマークｙオフセット値（単位：０．１画素）
out_inf[4] ・・・・・・・・・・・・・・ 登録時の左上端Ｘ座標
out_inf[5] ・・・・・・・・・・・・・・ 登録時の左上端Ｙ座標
out_inf[6] ・・・・・・・・・・・・・・ 登録時のサーチモード
（０：高速／１：黒線／２：白線／３：通常）
out_inf[7] ・・・・・・・・・・・・・・ Ｒｅｓｅｒｖｅｄ＝＝０
◎オフセット値は、登録時の左上端Ｘ，Ｙ座標からのオフセット

戻り値

処理結果

値	意　　　味
０	正常終了しました。
－１	パラメータで指定した名称の「サーチ・パタン」は登録されていないので処理ができません。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

登録パタンの縦横サイズを参照し、表示します。

```
#include "f_search.h"
#include "f_pinf.h"
#include "f_stdio.h"

void inf()
{
    char s[30];
    int  out_inf[8],name;

    name = Lib_get_gray_ptn_name(  );
    Lib_gs_infpat( name, out_inf );
    Lib_sprintf( s, "XSIZE:%d", out_inf[0] );
    Lib_chrdisp( 1, 1, s );
    Lib_sprintf( s, "YSIZE:%d", out_inf[1] );
    Lib_chrdisp( 1, 2, s );
}
```

留意事項

ありません。

Lib_gs_1dsppat

機　能　　パタン個別表示ライブラリ

形　式

```
#include ″f_search.h″
int Lib_gs_1dsppat( int p_name, int c_mode, int c_zoom,
                    int c_sx, int c_sy, int c_xofset, int c_yofset );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

登録済み「サーチ・パタン」の画像データを、原寸及び２倍、４倍または８倍に拡大してモニタＴＶに表示します。（同時にセンター・マークも線画表示されます）

① ***p_name*** はモニタＴＶ表示をするパタン名称です。
　パタン名称は Lib_gs_defpat,Lib_gs_usepat ライブラリで登録したものを使用します。

② ***c_mode*** は処理モードです。
　ゼロ・・・・・・・・・・　濃淡画像メモリ、文字フレームバッファ共にクリアーしてから③～⑥のパラメータに従って「サーチ・パタン」画像及びセンター・マークを表示します。
　ノンゼロ・・・・・・　③～⑦のパラメータについて１回前の本ライブラリ引用時の値とチェックして、必要に応じて濃淡画像メモリ、文字フレームバッファの再表示を行います。

③ ***c_zoom*** はサーチ・パタンを表示するときの倍率です。
　１，２，４，８，１６または３２の数値で指定します。

④ ***c_sx*** はサーチ・パタン表示Ｘ座標

⑤ ***c_sy*** はサーチ・パタン表示Ｙ座標
　「サーチ・パタン」の左上端を、モニタＴＶ座標系のどこに表示するか指定します。
　　［注］モニタＴＶへの表示位置は本指定値に上記の拡大率を乗じた位置となります。
　　　　（実際に表示されるモニタＴＶ座標位置は c_sx*c_zoom, c_sy*c_zoom となります）

⑥ ***c_xofset*** はセンター・マークＸオフセット

⑦ ***c_yofset*** はセンター・マークＹオフセット
　センター・マーク位置を、「サーチ・パタン」の左上端からのオフセットで指定します。
　（単位は０．１画素で、拡大率とは無関係に－９９９９～＋９９９９の数値で指定）

戻り値

処理結果

値	意　　　　味
０	正常終了しました。
－１	パラメータで指定した名称の「サーチ・パタン」は登録されていないので処理ができません。
－２	パラメータ・エラー（拡大率が範囲外）です。
－５	パラメータ・エラー（センター・マークＸ・Ｙオフセットが範囲外）です。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

登録パタンを表示し、位置を調整します。

```
#include "f_search.h"
#include "f_pinf.h"

void adjmark()
{
    int  out_inf[8],name, status;
    int  n_zoom, x_ofset, y_ofset, x_pos, y_pos, mode;

    name = Lib_get_gray_ptn_name(  );
    if( NORMAL_RETURN == Lib_gs_infpat( name, out_inf ))
    {
        x_ofset = out_inf[2];
        y_ofset = out_inf[3];
        n_zoom = 1;  x_pos = 100; y_pos = 150; mode = 0;
        for( status = ON; OFF != status ; )
        {
            Lib_gs_1dsppat( name, mode, n_zoom, x_pos,
                                          y_pos, x_ofset, y_ofset );
            mode = 0xff;
            /*必要に応じて各パラメータを更新*/
        }
    }
}
```

留意事項

○マスク部分の表示
　マスクの設定されている部分の画像データは、「黒」（＝濃度値０）で表示されます。

○システム・パラメータ処理ウィンドウ
　ＣＳＣ９０Ｘシリーズのシステムパラメータである処理ウィンドウは（0,0)～(511,479）に設定されます。

○画像メモリ
　「サーチ・パタン」の画像イメージデータが濃淡画像メモリに転送後表示されます。
　「サーチ・パタン」の外枠は、濃淡画像メモリに描画されます。
　センター・マーク・オフセット位置は、文字フレームバッファに表示されます。

○サーチ・パタン格納領域
　既に、登録済みの「サーチ・パタン」の画像イメージデータが表示の対象となります。

Lib_gs_upmark

機　能

センター・マーク更新ライブラリ

形　式

```
#include "f_search.h"
int Lib_gs_upmark( int p_name, int p_xofset, int p_yofset );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

既に登録されている「サーチ・パタン」の、センター・マーク・Ｘ／Ｙオフセット値をライブラリ引用時のパラメータ値で直接更新します。

① ***p_name*** はセンター・マークＸ／Ｙオフセットを更新するパタン名称です。
パタン名称は Lib_gs_defpat，Lib_gs_usepat ライブラリで登録したものを使用します。

② ***p_xofset*** はセンター・マークＸオフセットです。
サーチ実行時に返答される“Ｘ座標”は、「サーチ・パタン」の左上端に、ここで指定したオフセットを加算した座標位置となります。
指定可能な範囲は－９９９９～＋９９９９です。（単位：０．１画素）

③ ***p_yofset*** はセンター・マークＹオフセットです。
サーチ実行時に返答される“Ｙ座標”は、「サーチ・パタン」の左上端に、ここで指定したこのオフセットを加算した座標位置となります。
指定可能な範囲は－９９９９～＋９９９９です。（単位：０．１画素）

戻り値

処理結果

値	意　　味
０	正常終了しました。
－１	パラメータで指定した名称の「サーチ・パタン」は登録されていないので処理ができません。
－５	パラメータ・エラー（センター・マークＸ・Ｙオフセットが範囲外）です。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

登録パタンを登録後、センター・マーク・オフセット位置を更新します。

```
#include "f_search.h"
#include "f_pinf.h"
#define  XOFFSET  1255
#define  YOFFSET  2349

void adjmark()
{
    int  name;

    name = Lib_get_gray_ptn_name(  );
    if( NORMAL_RETURN == Lib_gs_defpat( name ))
    {
        Lib_gs_upmark( name, XOFFSET, YOFFSET );
    }
}
```

留意事項

○サーチ・パタン格納領域

前述のサーチ・パタン定義エリア指定 (Lib_gs_defadrs)ライブラリで指定された、「サーチ・パタン格納領域」に画像の相関計算に必要なデータが登録されます。

Lib_gs_u1param

機　能

サーチパタン表示ユーザ制御

形　式

```
#include "f_search.h"
int Lib_gs_u1param ( int uc_p1, int uc_p2, int uc_p3, int uc_p4, int uc_p5 );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

サーチ・パタン表示ライブラリを実行時、モニタＴＶ表示を見やすくするために背景色（グレイレベル）を変更するものです。

① ***uc_p1*** は個別表示クリアオプションです。
登録サーチ・パタン個別表示（Lib_gs_1dsppat）ライブラリ実行時、背景（フレームメモリ）を、クリアーするかどうかを指定します。
　ノンゼロ ‥‥‥‥‥ 背景のクリアーを実施します。（システム既定値）
　ゼロ ‥‥‥‥‥‥‥ 背景のクリアーを実施しません。

Lib_gs_1dsppat ライブラリを使用して、複数のパタンを同時に表示する場合に指定します。但し、ユーザプログラムで事前にフレームメモリのフリーズ・クリアーを実施しなければなりません。

② ***uc_p2*** はパタン表示背景色（ワード）です。
サーチ・パタンがモニタＴＶに表示される
- 登録サーチ・パタン表示（Lib_gs_dsppat）ライブラリ
- 登録サーチ・パタン個別表示（Lib_gs_1dsppat）ライブラリ
- マスク設定（Lib_gs_defmask）ライブラリ
- センター・マーク・オフセット位置調整（Lib_gs_adjmark）ライブラリ

上記４個のライブラリ実行時の、背景色（グレイレベル）を、０００（黒）～２５５（白）の数値で指定します。
なお、背景色のシステム既定値は下記のようになっています。
- Lib_gs_dsppat, Lib_gs_defmask ‥‥‥‥ １２８
- Lib_gs_1dsppat, Lib_gs_adjmark ‥‥‥‥ ０００

③ ***uc_p3*** はダミーパラメータです。ゼロを指定してください。

④ ***uc_p4*** はダミーパラメータです。ゼロを指定してください。

⑤ ***uc_p5*** はダミーパラメータです。ゼロを指定してください。

戻り値

処理結果

値	意　味
0	正常終了しました。
-999	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

背景色を中間色にし、登録パタンを表示します。

```
#include "f_search.h"
#include "f_pinf.h"
#define  MID_COLOR  128
#define  DUMMY      0

void adjmark()
{
    int  out_inf[8],name, status;
    int  n_zoom, x_ofset, y_ofset, x_pos, y_pos, mode;

    name = Lib_get_gray_ptn_name( );
    if( NORMAL_RETURN == Lib_gs_inpat( name, out_inf ))
    {
        x_ofset = out_inf[2];
        y_ofset = out_inf[3];
        n_zoom = 1;  x_pos = 100; y_pos = 150; mode = 0;
        Lib_gs_u1param( 0, MID_COLOR, DUMMY, DUMMY, DUMMY );
        for( status = ON; OFF != status ; )
        {
            Lib_gs_1dsppat( name, mode, n_zoom, x_pos,
                                                  y_pos, x_ofset, y_ofset );
            mode = 0xff;
            /*必要ﾆ応ｼﾞﾃ各ﾊﾟﾗﾒｰﾀｦ更新*/
        }
    }
}
```

留意事項

○システム既定値

サーチ・パタン定義(Lib_gs_defadrs)ライブラリ実行時（初期化モードにかかわらず）、
- ・個別表示クリアオプション
- ・パタン表示背景色

にはシステム既定値が設定されます。

Lib_gs_exdefpat

機　能

拡張画像時用ユーザ指定サーチ・パタン登録

形　式

```
#include "f_search.h"
int Lib_gs_exdefpat( int name, int xoffset, int yoffset, int x_scale, int y_scale );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

メモリ上の画像データをサーチの対象となる「サーチ・パタン」に登録します。
登録時のマウス操作は Lib_gs_defpat の項を参照してください。

① ***name*** はパタン名称です。

② ***x_offset*** はＸ方向オフセットです。

③ ***y_offset*** はＹ方向オフセットです。

④ ***x_scale*** はＸ方向縮尺です。

⑤ ***y_scale*** はＹ方向縮尺です。

戻り値

処理結果

値	定　　数	意　　味
０	ありません	正常終了しました。
－１	ありません	既に２００個登録済みです。
－２	ありません	格納領域不足のため登録できません。
１	ありません	キー操作のキャンセルにより処理を取り消しました。
-9999	ありません	Lib_gs_defadrsを未実行です。

例

縮小画像を対象にパタンを登録します。

```
#include "f_search.h"

int entry()
{
    int scale;

    scale = Lib_get_frame_ratio();

    return(Lib_gs_exdefpat ( Lib_get_gray_ptn_name(), 0, 0, Scale, Scale ) );
}
```

留意事項

○切り出し画像の場合は
　Ｘ始点オフセットに切り出しウインドウＸ始点を
　Ｙ始点オフセットに切り出しウインドウＹ始点を入れ、Ｘ方向縮尺・Ｙ方向縮尺は共に
　１にしてください。

○縮小画像の場合は
　Ｘ始点オフセットに０
　Ｙ始点オフセットに０入れ、Ｘ方向縮尺・Ｙ方向縮尺は共に Lib_get_frame_ratio の戻
　り値を入れてください。

Lib_gs_disp_result

機　能

拡張画像時用サーチ結果表示

形　式

```
#include "f_search.h"
int Lib_gs_disp_result( int name, int xoffset, int yoffset, int x_scale,
                        int y_scale, int number, int *result, int guif );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

ｍ×ｎの拡張画像メモリ時のグレイサーチ結果を表示します。

① ***name*** はパタン名称です。

② ***x_offset*** はＸ方向オフセットです。

③ ***y_offset*** はＹ方向オフセットです。

④ ***x_scale*** はＸ方向縮尺です。

⑤ ***y_scale*** はＹ方向縮尺です。

⑥ ***number*** はサーチした個数です。

⑦ ****result*** はサーチ結果格納ポインタです。

⑧ ***guif*** はパッド表示フラグです。

値	定　　数	意　　味
0	OFF	サーチ座標位置を表示し、戻ります。
1	ON	サーチ座標位置を表示し、システムのサーチ試行と同様なパッドを表示します。

戻り値

処理結果

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

サーチを実行し，結果を表示します。

```
#include "f_search.h"

#define  MAX_NO_GRAY_SEARCH                  50
static int Rslt_Buf[(MAX_NO_GRAY_SEARCH * 3)];

void search()
{
        int number;
        int scale;

        scale = Lib_get_frame_ratio();
        number = Lib_gs_xsearch ( DISAGREE_PTN, NON_DISPLAY,
                                  Lib_get_gray_ptn_name(),
                                  Lib_get_gray_search_no(),
                                  Lib_get_gray_search_mode(),
                                  Lib_get_gray_complex(),
                                  Lib_get_gray_mid_lower(),
                                  Lib_get_gray_last_lower(),
                                  Rslt_Buf );

        if ( 0 < number )
        {
            Lib_gs_disp_result( Lib_get_gray_ptn_name(),
                                0, 0, scale, scale, number, Rslt_Buf, ON );
        }
}
```

留意事項

○この関数をコールする前に Lib_gs_search の計測が成功していなくてはいけません。

○ Lib_gs_search の結果表示フラグは必ず，ＯＦＦにしてください。

○パッド表示フラグをＯＮにした場合は Lib_init_cursor を実行前にコールする必要があります。

○切り出し画像の場合は
Ｘ始点オフセットに切り出しウインドウＸ始点を
Ｙ始点オフセットに切り出しウインドウＹ始点を入れ、Ｘ方向縮尺・Ｙ方向縮尺は共に１にしてください。

○縮小画像の場合は
Ｘ始点オフセットに０
Ｙ始点オフセットに０入れ、Ｘ方向縮尺・Ｙ方向縮尺は共に Lib_get_frame_ratio の戻り値を入れてください。

Lib_gs_ptn_compare

機　能

パタンの比較

形　式

```
#include "f_search.h"
int Lib_gs_ptn_compare( int mem_no, int name, int xpos,
                        int ypos, int permit, int mode );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

グレイサーチの登録画像パタンと、指定メモリの指定位置の画像パタンを指定許容濃度差内かを判定します。
登録してあるサーチ画像は６ビットに圧縮されているので、８ビットに戻して比較を行います。
比較画像パタンは線形変換後、登録画像と比較しているので、画像の明暗に影響を受け難いですが、位置のズレは、エッジが影響を受けます。

① ***mem_no***は、比較画像パタンのメモリ番号です。（０～　）

② ***name*** は、パタン名称です。
　パタン名称は Lib_gs_defpat，Lib_gs_usepatライブラリで登録したものを使用します。

③ ***xpos*** は、比較画像のＸ始点（左上）です。

④ ***ypos*** は、比較画像のＹ始点（左上）です。

⑤ ***permit*** は、登録画像と比較画像の濃度許容差です。

⑥ ***mode*** は、欠陥部分の表示オプションです。

値	定　　数	意　　味
0	ＯＦＦ	表示しません。
1	ＯＮ	表示します。

戻り値

値	定　　数	意　　味
－９９９	ERROR_RETURN	異常終了しました。
０～	ありません	許容範囲外のドット数です。

例

サーチ後、パタンを比較します。

```
#include "f_search.h"
#include "f_gui.h"
static  int                 *y_wbase;        /*  ﾜｰﾄﾞ型ﾍﾞｰｽｱﾄﾞﾚｽ  */
#define Y_PTNNAME           *(y_wbase)       /*  +0 ﾊﾟﾀﾝ名称      */
#define Y_P_XOFSET          *(y_wbase +  3)  /*  +12 ｾﾝﾀｰ･ﾏｰｸxｵﾌｾｯﾄ*/
#define Y_P_YOFSET          *(y_wbase +  4)  /*  +16 ｾﾝﾀｰ･ﾏｰｸyｵﾌｾｯﾄ*/
#define  MAX_NO_GRAY_SEARCH  50

static int      Rslt_Buf[MAX_NO_GRAY_SEARCH][3];

void sample_execute( )
{
    int rtn_code;
    int x, y;
    int max_val;
    char  buff[30];
    int count,mem_no;

    Lib_init_cursor();
    mem_no     = Lib_get_gray_memory();

    y_wbase = (int *)Lib_gs_ptn_get( Lib_get_gray_ptn_name(  ) );
                                                     /*登録画像ﾉｱﾄﾞﾚｽ*/
    rtn_code = Lib_gs_psearch ( ON_DISPLAY, Lib_get_gsearch_no(  ),
                                                  (int *) Rslt_Buf);

    if( 0 >= rtn_code )
    {
        Lib_sprintf( buff, "%s%d","ﾘﾀｰﾝ･ｺｰﾄﾞ:",rtn_code );
        Lib_display_message(100, 200,"ｴﾗｰ", buff );
    }
    else
    {
         x = (Rslt_Buf[0][0]  / 10) - (Y_P_XOFSET / 10);
         y = (Rslt_Buf[0][1]  / 10) - (Y_P_YOFSET / 10);
         count = Lib_gs_ptn_compare( mem_no, name, x, y, 50, 1 );
         Lib_sprintf( buff,"%s:%4d","欠陥ﾄﾞｯﾄ"count);
         Lib_display_comment( 380, 1, buff );
    }
}
```

留意事項

○表示オプションの表示をＯＮにした場合、処理時間が掛かります。

Lib_gs_open_data_file

機　能

システム共通データのオープン

形　式

```
#include "f_search.h"
int Lib_gs_open_data_file( int mode );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	

解　説

システム用グレイサーチ・パタンデータ・ファイルをディスクから検索し、そのファイルが存在する場合はファイルからメインメモリにロードします。
存在しない場合はメインメモリにファイルを確保するだけです。
そのときの確保容量はシステムパラメータのグレイサーチ・パタンデータ・ファイル容量となります。
ファイルが正常に確保された場合、Lib_gs_defadrs（定義エリア指定）を実行します。
従って Lib_gs_defadrs は改めて実行する必要はありません。

① ***mode*** は初期化モードです。

値	定　　数	意　　味
0	INITIAL_PTN_AREA	格納エリアを初期化します。
1	CONTINUE_PTN_AREA	格納エリアを継続して使用します。

戻り値

処理結果

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

システム共通グレイサーチ・パタンデータ・ファイルを継続オープンします。

```
#include "f_search.h"

void gs_start()
{
    Lib_gs_open_data_file( CONTINUE_PTN_AREA );

    Lib_gs_window ( 0xff, -1, 0, 0, 0 );
}
```

留意事項

ありません。

Lib_gs_save_data_file

機　能

システム共通データのセーブ

形　式

```
#include "f_search.h"
int Lib_gs_save_data_file( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	

解　説

システム用グレイサーチ・パタンデータ・ファイルをセーブします。
グレイサーチの処理は継続して行います。

戻り値

処理結果

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

システム共通グレイサーチ・パタンデータ・ファイルをセーブします。

```
#include "f_search.h"

void gs_end()
{
    Lib_gs_save_data_file();
}
```

留意事項

ありません。

Lib_gs_close_data_file

機　能

システム共通データのクローズ

形　式

```
#include "f_search.h"
int Lib_gs_close_data_file( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	

解　説

システム用グレイサーチ・パタンデータ・ファイルをクローズします。
クローズするという事はメインメモリファイルを削除する事です。
システム共通グレイサーチ・パタンデータ・ファイルを保存する場合は、必ず前項の
Lib_gs_save_data_file を実行してください。

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

システム共通グレイサーチ・パタンデータ・ファイルをクローズします。

```
#include "f_search.h"

void gs_end()
{
    Lib_gs_close_data_file( );
}
```

留意事項

ありません。

Lib_gs_get_data_file_adrs

機　能　　システム共通データのアドレス参照

形　式

```
#include "f_search.h"
char *Lib_gs_get_data_file_adrs( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	

解　説　　システム共通グレイサーチ・パタンデータ・ファイルのアドレスを参照します。

戻り値　　システム共通グレイサーチ・パタンデータ・ファイルのポインタを返します。

例　　登録済みサーチ・パタン個数を参照します。

```
#include "f_search.h"
#include "f_gui.h"

static  int       *base;          /*ﾜｰﾄﾞ型ﾍﾞｰｽｱﾄﾞﾚｽ          */
#define PTN_NO    *(base +  1)    /*+4 登録済ｻｰﾁ･ﾊﾟﾀﾝ個数   */

int get_ptn_no()
{
    base = (int *)Lib_gs_get_data_file_adrs(  );
    return( PTN_NO );
}
```

留意事項　　ありません。

Lib_gs_usepat_square

機　能

ユーザ指定サーチ・パタン（センターマーク自動更新）四角形登録用

形　式

```
#include "f_search.h"
int Lib_gs_usepat_square( int p_name, int p_sx, int p_sy,
                          int p_ex, int p_ey, int margin );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

パラメータで指定された四角形（正方形及び長方形）形状の、中心位置をセンターマークとして自動的に算出して「サーチ・パタン」に登録します。

① ***p_name*** はパタン名称です。名称は４文字の null 以外を付けてください。
（登録するサーチ・パタンに付与する名前）
［注］ここで設定した名称で、サーチ実行（＝Lib_gs_search）ライブラリでサーチするパタンを指定します。

② ***p_sx*** は「サーチ・パタン」に登録する画像の左上端Ｘ座標です。
範囲は２０～４８０の数値で指定します。ただし、ＸＳ≦４９２－Ｗ

③ ***p_sy*** は「サーチ・パタン」に登録する画像の左上端Ｙ座標です。
範囲は２０～３９６の数値で指定します。ただし、ＸＹ≦４６０－Ｈ

④ ***p_ex*** は「サーチ・パタン」に登録する画像の右下端Ｘ座標です。

⑤ ***p_ey*** は「サーチ・パタン」に登録する画像の右下端ｙ座標です。

⑥ ***margin*** は登録ワークサイズからマージンの比率分のパタン登録を行います。
指定範囲１～１００（％）の数値で指定します。

戻り値

処理結果

値	定　　数	意　　味
０	ありません。	登録が正常終了しました。
－１	ありません。	パラメータエラー。
－２	ありません。	ワークメモリが取得できません。

例

パタン座標　始点（100,100）　終点（300,300）　マージン３０（%）の”ＡＢＣＤ”からなる名称の「サーチ・パタン」を登録します。

```
#include "f_search.h"
#include "f_image.h"
#include "f_video.h"
#include "f_gui.h"


int set_pnt()
{
    int name;

    name = 0x44434241; /*ABCD*/

    Lib_freeze( TRANSMIT );

    Lib_gs_usepat_square( name, 100, 100, 300, 300, 30 );

    Lib_display_keyinput( 10, 10, "wait" );
    Lib_memory_clear( LINE_PLANE );

    Lib_freerun();
}
```

留意事項

○ Lib_gs_usepat，Lib_gs_fremask を参照

○確認用に抽出された四角形の外周エッジ、及び算出されたセンターマーク位置が線画用フレーム・バッファに表示されます。

○実際に登録されるパタンのサイズは上下左右を縦横サイズのマージン分拡げたサイズとなります。

$$登録縦サイズ = \left(p_ey - p_sy \right) * \left(1 + margin / 100 * 2 \right)$$

○センターマークの算出は、パラメータで指定された短形の周内部で抽出したエッジにより、パタン外周部の近似直線を求めることにより行われます。

Lib_gs_usepat_circle

機　能

ユーザ指定サーチ・パタン（マスク自動設定&センターマーク自動更新)円形登録用

形　式

```
#include "f_search.h"
int Lib_gs_usepat_circle( int p_name, int x_center, int y_center, int radius );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	

解　説

パラメータで指定された円形パタンを「サーチ・パタン」に登録します。
センターマークは円形パタンの外周エッジを抽出し、最小自乗法円近似により求めた中心位置が登録されます。

① ***p_name*** はパタン名称です。名称は４文字の null 以外を付けてください。
（登録するサーチ・パタンに付与する名前）
[注]ここで設定した名称で、サーチ実行（=Lib_gs_search)ライブラリでサーチするパタンを指定します。

② ***x_center*** は「サーチ・パタン」に登録する画像の中心Ｘ座標です。

③ ***y_center*** は「サーチ・パタン」に登録する画像の中心Ｙ座標です。

④ ***radius*** は「サーチ・パタン」に登録する画像の半径サイズです。

戻り値

処理結果

値	定　　数	意　　味
０	ありません。	登録が正常終了しました。
－１	ありません。	パタン名登録エラー。
－２	ありません。	ワークメモリが取得できません。
－３	ありません。	センターマークの検出に失敗しました。 （エッジがとれる画像にしてください）

例

パタン座標　中心(100,100)　半径６０（画素)の”ＡＢＣＤ”からなる名称の「サーチ・パタン」を登録します。

```
#include "f_search.h"
#include "f_image.h"
#include "f_video.h"
#include "f_gui.h"

int set_pnt()
{
    int name;

    name = 0x44434241; /*ABCD*/

    Lib_freeze( TRANSMIT );

    Lib_gs_usepat_circle( name, 100, 100, 60 );

    Lib_display_keyinput( 10, 10, "wait" );
    Lib_memory_clear( LINE_PLANE );
    Lib_freerun();
}
```

留意事項

○ Lib_gs_usepat, Lib_gs_fremask を参照してください。

○確認用に抽出された円形の外周エッジ、及び算出されたセンターマーク位置が線画用フレーム・バッファに表示されます。

○パラメータで指定される円と、実際の円形パタンは５画素程度以上の余裕を持たせてください。（パラメータの円形領域の内部に実際のパタンが入るようにしてください。）

○実際に登録されるパタンのサイズは、パラメータで指定される円に外接する短形となり、余分な部分は自動的にマスクされて登録します。

Lib_gs_open

機　能

グレイサーチライブラリのオープン

形　式

```
#include "f_search.h"
int Lib_gs_open( int max_fxsize, int max_fysize, int max_objnum, int dummy );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
				○	

解　説

グレイサーチ実行時に使用する圧縮画像バッファ、結果格納バッファを確保します。
サーチ実行時にエラーコード －1004 が返ってきた場合、またはサーチ個数の上限を増やしたい場合は、本ライブラリで結果格納バッファを増やしてください。
また、現在のキャプチャサイズより大きな画像メモリをサーチ対象とする場合は、圧縮画像バッファも大きくする必要があります。
特に上記のような理由に当てはまらない場合には、本関数をコールする必要はありません。

① ***max_fxsize*** は圧縮画像バッファの横方向画素数の最大値です。

② ***max_fysize*** は圧縮画像バッファの縦方向画素数の最大値です。

③ ***max_objnum*** は結果格納バッファに格納する候補点数の最大値です。(デフォルト400)

④ ***dummy*** はダミーです。必ず－1を指定してください。

戻り値

処理結果

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

プログラム起動時に結果格納バッファを1000にします。

```
#include "f_search.h"

int main( void )
{

    Lib_gs_open( Lib_get_fx_size(), Lib_get_fy_size(), 1000, -1 );
    /***** 何らかの処理 *****/

}
```

留意事項

○圧縮画像バッファの縦、横方向画素数の最大値は、サーチ対象となる画像メモリの縦、横サイズの最大値を指定してください。

Lib_gs_get_ptnname

機　能

登録順によるパタン名の取得

形　式

```
#include "f_search.h"
int Lib_gs_get_ptnname( int index, int *ptn_name );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
				○	

解　説

登録済みグレイサーチパタンの中から、index+1番目に登録されたパタンの名称を取得します。

① ***index*** はパタン名を取得するパタンの登録順です。(0から始まります)

② ****ptn_name*** はパタン名を格納するバッファへのポインタです。

戻り値

処理結果

値	意　　味
0	正常終了しました。
－１	指定されたパタンは登録されていません。
－９９９	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

パタン個数を取得し、最後に登録したパタンの名称を取得します。

```
#include "f_search.h"

int func( void )
{
    int     ptn_name;

    Lib_gs_get_ptnname( Lib_gs_get_ptnnum()-1, &ptn_name );

    return( ptn_name );
}
```

留意事項

ありません。

Lib_gs_get_ptnnum

機　能

パタン個数の取得

形　式

```
#include "f_search.h"
int Lib_gs_get_ptnnum( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
				○	

解　説

登録済みグレイサーチパタンの個数を取得します。

戻り値

処理結果

値	意　　味
０以上	パタン個数です。
－９９９	サーチ・パタン定義エリア指定(=Lib_gs_defadrs)ライブラリが実行されていません。

例

登録順によるパタン名の取得（Lib_gs_get_ptnname）を参照してください。

留意事項

ありません。

Lib_gs_get_ptnfile_size

機　能

パタン定義エリアサイズ取得

形　式

```
#include "f_search.h"
int Lib_gs_get_ptnfile_size( char *p_adrs );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
				○	

解　説

登録済みグレイサーチパタンの定義エリアサイズを取得します。

戻り値

処理結果

値	意　　味
０以上	パタン定義エリアサイズ
－1	エラー(Lib_gs_defadrs 未実行もしくはポインタが不正)

例

```
#include "f_search.h"

int get_ptnfile_size( char *p_adrs )
{
    return Lib_gs_get_ptnfile_size( p_adrs );
}
```

留意事項

ありません。

３．Ｓ回転サーチライブラリ

本ライブラリは、あらかじめ登録しておいたサーチパタンの位置を探し出す（以降「サーチ」として記述）もので、サーチパタンのスケール変化／回転にも対応しています。
なお、サーチパタンの形状は、線分で構成されたものに限定されます。

本ライブラリの特長として次が挙げられます：

○回転しているパタンを高速にサーチします。
さらにスケールが変化する場合でもサーチ時間にほとんど変動はありません。

○精サーチ実行によって高精度のサーチが実現できます。

○パタン形状から自動的に特徴量を抽出しますので、面倒なパラメータの設定はありません(サーチに必要な条件設定のみです)。

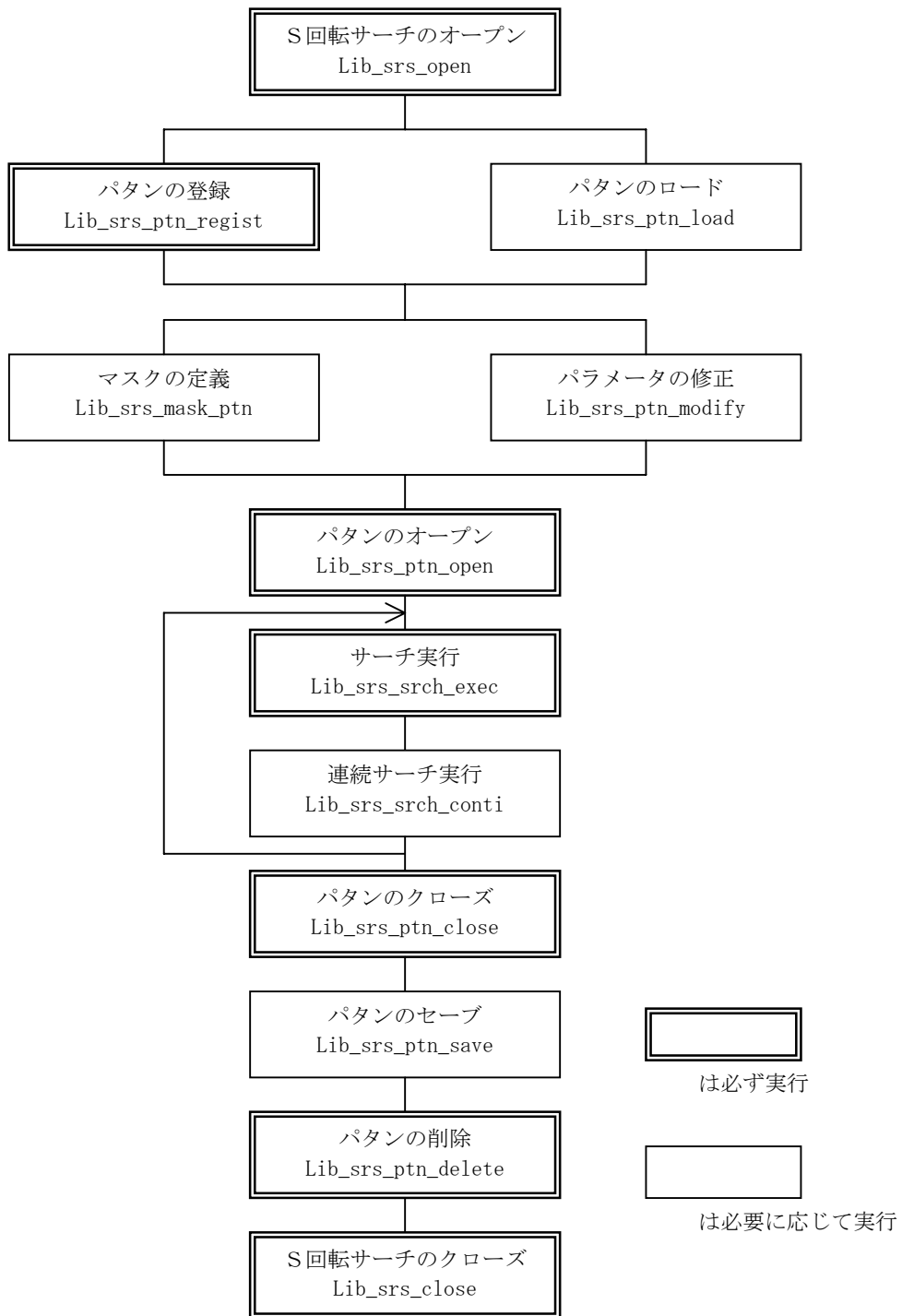
○１枚の画像から異種・複数個数のパタンをサーチさせる場合でも処理時間にほとんど変動はありません(ただし、粗サーチのみ実行の場合)。

<<<注意>>>

○サーチパタンには安定した線分の組が幾つか必要です。
例えば、針のように非常に鋭角に交わる２線分しかないようなパタンをサーチすることはできません(結果が保証できないため、サーチできないようにしています)。

○本ライブラリは９０２シリーズ専用です。９０１、９０３、９０４シリーズでは使用できません。

○高分解能カメラには対応していません。
通常のエリアカメラ(分解能５１２×４８０画素)にのみ対応しています。

【Ｓ回転サーチの処理手順】

Lib_srs_open

機　能

Ｓ回転サーチのオープン

形　式

```
#include "f_srs.h"
int Lib_srs_open( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

Ｓ回転サーチを実行するのに必要な初期化を行ないます。

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-101	SRSERR_MEMORY	メモリ不足のための異常終了です。
-108	SRSERR_DUPLIC	Ｓ回転サーチを２重にオープンしようとしたための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合には本ライブラリを最初にコールし、正常終了されなければなりません。また、本ライブラリを複数回コールすることはできません。

○Ｓ回転サーチで使用したワークメモリを解放するため、Ｓ回転サーチを終了する際には、次頁のクローズ Lib_srs_close を必ずコールしてください。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_close

機　能

Ｓ回転サーチのクローズ

形　式

```
#include "f_srs.h"
Lib_srs_close( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

Ｓ回転サーチのクローズ処理(メモリ解放等)を行ないます。

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○Ｓ回転サーチで使用したワークメモリを解放するため、Ｓ回転サーチを終了する際には、本ライブラリを必ずコールしてください。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_ptn_regist

機　能

サーチパタンの登録

形　式

```
#include "f_srs.h"
int Lib_srs_ptn_regist( int ptn_name, int mem_no, int sx, int sy,
                        int ex, int ey, DPNT2_T std_pnt,
                        int st_q, int ed_q, int min_s, int max_s );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

Ｓ回転サーチのサーチパタンを登録します。

① ***ptn_name*** は登録するパタンの名称です。このパタンに固有のものを与えてください。

② ***mem_no*** は登録したいパタンが存在する濃淡画像のメモリの番号です。

③～⑥ ***sx***, ***sy***, ***ex***, ***ey*** は②で指定した濃淡画像内で登録パタンが存在する矩形領域を指定するパラメータです。矩形領域の左上の点の座標を(sx, sy)とし、右下の点の座標を(ex, ey)とします。

⑦ ***std_pnt*** は回答基準点(通知点)です。サーチの結果得られる回答位置はここで指定した点に対応する座標で与えられます。std_pnt のｘ座標、ｙ座標は③～⑥で指定した矩形領域の左上の点(sx, sy)からのオフセット値で与えてください。構造体 DPNT2_T は２次元平面上の点の座標を浮動小数点で表わすためのもので、f_srs.h で次のように定義されています。

```
typedef struct
{
        double  x;      /*  x 座標  */
        double  y;      /*  y 座標  */
}DPNT2_T;
```

⑧, ⑨ ***st_q***, ***ed_q*** は登録するパタンをサーチする際の回転角の範囲で、単位は｢度｣です。サーチ画面の中では登録パタンがこの範囲で回転しているものとしてサーチを実行します。
回転角の範囲は次式を満たすように指定されなければなりません。

$$-360 \leqq st_q \leqq 360, \quad 0 \leqq ed_q \leqq 360$$

$$st_q \leqq ed_q, \quad ed_q - st_q \leqq 360$$

⑩, ⑪ ***min_s***, ***max_s*** は登録するパタンをサーチする際のスケールの範囲です。
スケールの単位は「%」です。サーチ画面の中では登録パタンがこの範囲でスケール変化しているものとしてサーチを実行します。
スケールの範囲は次式を満たすように指定されなければなりません。

$$0 < min_s \leqq max_s$$

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-101	SRSERR_MEMORY	メモリ不足のための異常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-103	SRSERR_PARAM	入力パラメータが不適切なための異常終了です。
-108	SRSERR_DUPLIC	同名の登録パタンが既にあるための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○名称の異なる複数のパタンを登録する事ができます。

○登録したパタンをサーチさせるためにはパタンのオープン Lib_srs_ptn_open をコールしなければなりません。

○登録する矩形領域はワークの全体、一部分のどちらでも構いません。

○登録したいワークが小さすぎる場合は登録出来ない場合があります。
５０×５０以上のパタンを登録することをお勧めします。

○スケールの範囲の指定で、min_s = max_s とした場合とそうでない場合、サーチで実行されるアルゴリズムに若干の違いがあります。また、不必要にスケールの範囲を大きく設定すると思わぬ誤サーチを起こしてしまう恐れがあります。安定したサーチを実行するために、スケールの範囲は必要最小限に設定してください。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_ptn_delete

機　能

登録済みサーチパタンの削除

形　式

```
#include "f_srs.h"
int Lib_srs_ptn_delete( int ptn_name );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

登録されているパタンを削除します。

① ***ptn_name*** は削除したい登録パタンの名称です。

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-105	SRSERR_PATTERN	指定パタンが未登録のための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○既にパタンオープンされている登録パタンを削除する場合にはパタンクローズ Lib_srs_ptn_close をコールしてから削除してください。

○Ｓ回転サーチを終了する際には、使用したワークメモリを解放するために全ての登録パタンを本ライブラリによって削除し、さらにクローズ Lib_srs_close によってクローズ処理を行なってください。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_mask_define

機　能　　登録パタンのマスクの定義

形　式

```
#include "f_srs.h"
int Lib_srs_mask_define( int ptn_name, char mask[] );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

登録パタンに対して、マスク(サーチの不感帯)を定義します。
マスクされた画素にある情報はサーチ実行時に使用されなくなります。

① ***ptn_name*** はマスクを定義する登録済みパタンの名称です。

② ***mask[]*** は定義したいマスク情報です。
このバッファはパタンの横画素数×縦画素数のサイズの配列でなければなりません。
マスクをかける(オンにする)画素には１を、かけない(オフにする)画素には０を設定します。

mask[ ] の各要素と登録パタンの画素との関係は下図のようになっています。

（０，０）	（１，０）	（２，０）
（０，１）	（１，１）	（２，１）
（０，２）	（１，２）	（２，２）
（０，３）	（１，３）	（２，３）

横サイズ３、縦サイズ４の登録パタン

mask[]の先頭アドレスからのオフセット

オフセット	
0	（０，０）のマスク（０ or １）
1	（１，０）のマスク
2	（２，０）のマスク
3	（０，１）のマスク
4	（１，１）のマスク
	⋮
１１	（２，３）のマスク

mask[]……サイズは３×４size of （char)＝１２

すなわち、横サイズがＳの登録パタンの画素（ｘ、ｙ）(ただし、ｘ、ｙは左上からのオフセット)にマスクをかけたい場合は

$$mask\left[\ x + y \times S\ \right] = 1$$

とし、マスクをかけたくない場合には

$$mask\left[\ x + y \times S\ \right] = 0$$

としてください。

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-101	SRSERR_MEMORY	メモリ不足のための異常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-105	SRSERR_PATTERN	指定パタンが未登録のための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○本ライブラリで定義したマスクは Lib_srs_ptn_open をコールすることによって初めてサーチに反映されるようになります。
したがって、Lib_srs_ptn_open によって既にオープンされているパタンを本ライブラリで修正する場合には、定義されたマスクをサーチに反映させるために次のようにしてください；

・Lib_srs_ptn_close によって一旦パタンをクローズする
・本ライブラリによってマスクを定義する
・Lib_srs_ptn_open によって再度パタンをオープンする

○指定された登録パタンに既にマスクが定義されている場合、本ライブラリをコールすることによって　以前のマスク情報は更新されます。

○登録パタンのサイズは Lib_srs_get_ptn_image_size で取得できます。また、既に定義されている　マスクの情報は Lib_srs_get_mask_ptn で取得できます。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_ptn_modify

機　能

登録パタンのパラメータの一部修正

形　式

```
#include "f_srs.h"
int Lib_srs_ptn_modify( int ptn_name, DPNT2_T std_pnt,
                        int st_q, int ed_q, int min_s, int max_s );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

登録パタンのパラメータの一部を修正します。

① ***ptn_name*** は修正したい登録パタンの名称です。

②～⑥ ***std_pnt***, ***st_q***, ***ed_q***, ***min_s***, ***max_s*** はパラメータの修正値です。
　各パラメータの意味については Lib_srs_ptn_regist の解説を参照してください。

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-103	SRSERR_PARAM	入力パラメータが不適切なための異常終了
-105	SRSERR_PATTERN	指定パタンが未登録のための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○本ライブラリで修正した情報は Lib_srs_ptn_open をコールすることによって初めてサーチに反映されるようになります。
したがって、Lib_srs_ptn_open によって既にオープンされているパタンを本ライブラリで修正する場合には、修正した情報をサーチに反映させるために次のようにしてください。

・Lib_srs_ptn_close によって一旦パタンをクローズする
・本ライブラリによってパラメータを修正する
・Lib_srs_ptn_open によって再度パタンをオープンする

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_ptn_load

機　能

登録パタンをファイルからロード

形　式

```
#include "f_srs.h"
int Lib_srs_ptn_load( char file_name[] );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

登録パタンがセーブされているファイルをメインメモリへロードします。

① ***file_name[]*** はロードしたいパタン格納ファイルの名前です。例えば以下のように記述してください。

「ファイル "srs.ptn" をロードしたい場合」

```
Lib_srs_ptn_load( "srs.ptn" );
```

戻り値

値	定　　数	意　　味
０＜	ありません。	ロードされたファイルに格納されているサーチパタンの数です。
-101	SRSERR_MEMORY	メモリ不足のための異常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-106	SRSERR_FILE	指定ファイルがないか、パタンのファイルでないための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○登録パタンがメインメモリ上に存在する場合、本ライブラリでロードを実行するとこれらの登録パタンは削除されます(これに伴うパタンのクローズ処理、削除処理は自動的に行われます)。
ただし、指定ファイルが存在しないためにロードが異常終了した場合は上記の登録パタンは更新されずに残ります。

○ロードされたサーチパタンは自動的に登録処理されますので、Lib_srs_ptn_regist をコールする必要はありません。また、ロード以後は Lib_srs_ptn_regist にて登録したものと全く同等に扱うことが出来ます。

○ロードの対象となるのは Lib_srs_ptn_save によってディレクトリＦＳ０にセーブされたものに限ります。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_ptn_save

機　能

登録パタンをファイルにセーブ

形　式

```
#include "f_srs.h"
int Lib_srs_ptn_save( char file_name[] );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

メインメモリ上に存在するＳ回転サーチの全登録パタンを指定されたファイル名でセーブします。

① ***file_name[]*** はセーブしたいパタン格納ファイルに付ける名前です。
例えば以下のように記述してください。

「ファイルを "srs.ptn" でセーブしたい場合」

```
Lib_srs_ptn_save( "srs.ptn" );
```

戻り値

値	定　　数	意　　味
０＜	ありません。	セーブされたファイルのサイズ(バイト)です。
-101	SRSERR_MEMORY	メモリ不足のための異常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-105	SRSERR_PATTERN	パタンが１つも登録されていないのための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○ディレクトリＦＳ０にセーブされます。

○ＦＳ０に既に同名のファイルが存在する場合には上書きでセーブが実行されます。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_ptn_open

機　能

登録パタンのオープン

形　式

```
#include "f_srs.h"
void *Lib_srs_ptn_open( int ptn_name, int *err_rprt );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

登録パタンをサーチ実行可能なパタンにします(このの作業をパタンオープンといい、サーチ実行可能になったパタンをオープンパタンといいます)。

① ***ptn_name*** はパタンオープンしたい登録パタンの名称です。

② ****err_rprt*** はエラー報告です。

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-101	SRSERR_MEMORY	メモリ不足のための異常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-104	SRSERR_BEYOND	Ｓ回転サーチでサーチするにはパタンの情報が不十分なためにオープンできない異常終了です。
-105	SRSERR_PATTERN	指定パタンが未登録のための異常終了です。
-108	SRSERR_DUPLIC	指定パタンが既にオープンされているための異常終了です。

戻り値

値	定　　数	意　　味
0	NULL	異常終了です。エラー内容は上記のいずれかです。
０以外	ありません。	識別子です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○本ライブラリによってオープンされたパタンはＳ回転サーチを終了する前までに Lib_srs_ptn_close によってパタンのクローズ処理を行なってください。さもないと使用したワークメモリが解放されません。

○本ライブラリの返値として得られる識別子はオープンパタン固有に割り当てられるものであり、サーチ実行時やパタンのクローズの際のオープンパタンの引数として用いられます。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_ptn_close

機　能

オープンパタンのクローズ処理

形　式

```
#include "f_srs.h"
int Lib_srs_ptn_close( void *ptn );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

オープンパタンのクローズ処理を行ないます。
これ以降、このパタンをサーチすることは出来なくなります。

① ***ptn** はオープンパタンの識別子です。

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-105	SRSERR_PATTERN	指定パタンがオープンされていないのための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○パタンの削除とパタンのクローズとは別の処理です。Ｓ回転サーチを終了する際には全オープンパタンをクローズし、かつ削除しなければなりません。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_srch_exec

機　能　　Ｓ回転サーチの実行

形　式

```
#include ″f_srs.h″
int Lib_srs_srch_exec( void *ptn, int rq_srch_num, int mem_no,
                        int sx, int sy, int ex, int ey,
                              int thres, SRS_ANS_T ans[] );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説　　Ｓ回転サーチを実行します。

① ***ptn*** はパタンオープン Lib_srs_ptn_open で得られるオープンパタンの識別子です。この識別子を持つパタンをサーチします。

② ***rq_srch_num*** はサーチしたい個数です。

③ ***mem_no*** はサーチの対象となる濃淡画像メモリ番号です。

④～⑦ ***sx***, ***sy***, ***ex***, ***ey*** はサーチ範囲を表わすパラメータです。サーチ範囲は矩形領域で、左上の点の座標を(sx, sy)とし、右下の点の座標を(ex, ey)とします。

⑧ ***thres*** は回答のスコアのしきい値です。回答のスコアは１００点満点です。このスコアよりも低いスコアの回答は出力されません。次を満たすように設定してください。

$$0 < thres \leqq 100$$

⑨ ***ans[]*** はサーチ結果を格納するバッファです。
構造体 SRS_ANS_T は f_srs.h で次のように定義されています。

```
typedef struct
{
    double   x;         /*  x座標位置 */
    double   y;         /*  y座標位置 */
    double   q;         /*  回転角     */
    double   scale;     /*  スケール   */
    int      score;     /*  スコア     */
}SRS_ANS_T;
```

戻り値

値	定　　数	意　　味
-101	SRSERR_MEMORY	メモリ不足のための異常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-103	SRSERR_PARAM	入力パラメータが不適切なための異常終了です。
-105	SRSERR_PATTERN	指定パタンがオープンされていないのための異常終了です。
-109	SRSERR_SEARCH	サーチ画像に十分な情報がないかメモリ不足のための異常終了です。
0≦	ありません	正常終了で、サーチできた個数です。0の場合は見つからなかった、ということになります。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○回答の回転角ｑは -180.0 ＜ ｑ ≦ 180.0 で与えられます。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_srch_conti

機　能　　同一画面に対するＳ回転サーチの連続実行

形　式

```
#include "f_srs.h"
int Lib_srs_srch_conti( void *ptn, int rq_srch_num, int thres, SRS_ANS_T ans[] );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

あるオープンパタンを Lib_srs_srch_exec によってサーチした後、引き続き同じサーチ画像から次のオープンパタンをサーチする場合のサーチ実行ライブラリです。
Lib_srs_srch_exec によって取得できている画像情報を再利用しますので、本ライブラリでのサーチは非常に高速に実行されます。

① ***ptn** はパタンオープン Lib_srs_ptn_open で得られるオープンパタンの識別子です。この識別子を持つパタンをサーチします。

② ***rq_srch_num*** はサーチしたい個数です。

③ ***thres*** は回答のスコアのしきい値です。回答のスコアは１００点満点です。このスコアよりも低いスコアの回答は出力されません。次を満たすように設定してください。

$$0 < thres \leqq 100$$

④ ***ans[]*** はサーチ結果を格納するバッファです。構造体 SRS_ANS_T は f_srs.h で次のように定義されています。

```
typedef struct
{
    double    x;          /*  x座標位置 */
    double    y;          /*  y座標位置 */
    double    q;          /*  回転角     */
    double    scale;      /*  スケール   */
    int       score;      /*  スコア     */
}SRS_ANS_T;
```

戻り値

値	定　　数	意　　味
-101	SRSERR_MEMORY	メモリ不足のための異常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-103	SRSERR_PARAM	入力パラメータが不適切なための異常終了です。
-105	SRSERR_PATTERN	指定パタンがオープンされていないのための異常終了です。
-109	SRSERR_SEARCH	サーチ画像に十分な情報がないかメモリ不足のための異常終了です。
-110	SRSERR_IMAGE	サーチ画像がないための異常終了です(直前にLib_srs_srch_exec が実行されていません)。
0≦	ありません	正常終了で、サーチできた個数です。０の場合は見つからなかった、ということになります。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○回答の回転角 q は -180.0 ＜ q ≦ 180.0 で与えられます。

○Lib_srs_srch_exec でサーチを１回実行した後でのみ、本ライブラリを使用することができます。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_get_rgst_ptn_num

機　能

登録パタンの数の取得

形　式

```
#include "f_srs.h"
int Lib_srs_get_rgst_ptn_num( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

現在登録されているパタンの総数を取得します。

戻り値

値	定　数	意　味
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
0 ≦	ありません。	現在の登録パタンの総数です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_get_rgst_ptn_names

機　能

全登録パタンの名称の取得

形　式

```
#include "f_srs.h"
int Lib_srs_get_rgst_ptn_names( int names[] );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

現在登録されているすべてのパタンの名称を取得します。

① ***names[ ]*** はパタンの名称を格納するためのバッファです。
　必要なサイズは sizeof(int)×(全登録パタン数) です。

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○名称を格納するバッファは呼出し側で確保してください。
　その登録パタンの総数は Lib_srs_get_rgst_ptn_num で取得できます。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_get_ptn_image_size

機　能

登録パタンの画像のサイズを取得

形　式

```
#include "f_srs.h"
int Lib_srs_get_ptn_image_size( int ptn_name, int *size_x, int *size_y );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

登録パタンの画像の縦・横のサイズを取得します。

① ***ptn_name*** はサイズを取得したい登録パタンの名称です。

② ****size_x*** は登録パタンの画像の横サイズ(画素数)です。

③ ****size_y*** は登録パタンの画像の縦サイズ(画素数)です。

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-105	SRSERR_PATTERN	指定パタンが未登録のための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_get_ptn_image

機　能

登録パタンの画像を取得

形　式

```
#include "f_srs.h"
Lib_srs_get_ptn_image( int ptn_name, unsigned char *ptn_image );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

登録パタンの濃淡画像を取得します。連続領域バッファ *ptn_image に各画素の８ビットの輝度値が順に格納されます。順序は左上の画素から右に進み、最後は右下の画素にくる順序です（Lib_srs_mask_define の図の左側に描いてある矢印の順序)。

① ***ptn_name*** は画像を取得したい登録パタンの名称です。

② ****ptn_image*** は画像を格納するバッファです。
　必要なサイズは（パタン画像の横サイズ)×(パタン画像の縦サイズ）です。

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-105	SRSERR_PATTERN	指定パタンが未登録のための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○画像を格納するバッファは呼出し側で確保してください。その画像の縦・横のサイズは Lib_srs_get_ptn_image_size で取得できます。

○本ライブラリでは登録パタンのマスク情報は取得できません。マスク情報取得のためには Lib_srs_get_mask_ptn をご使用ください。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_get_ptn_param

機　能

登録パタンのパラメータの取得

形　式

```
#include "f_srs.h"
int Lib_srs_get_ptn_param( int ptn_name, DPNT2_T *std_pnt, int *st_q,
                           int *ed_q, int *min_s, int *max_s );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

パタンの登録 Lib_srs_ptn_regist 等で設定したパラメータを取得します。

① ***ptn_name*** はパラメータを取得したい登録パタンの名称です。

②～⑥ ****std_pnt***, ****st_q***, ****ed_q***, ****min_s***, ****max_s*** は取得されたパラメータです。
各パラメータの意味については Lib_srs_ptn_regist の解説を参照してください。

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-105	SRSERR_PATTERN	指定パタンが未登録のための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_get_mask_ptn

機　能

登録パタンのマスク情報の取得

形　式

```
#include "f_srs.h"
int Lib_srs_get_mask_ptn( int ptn_name, char *mask );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

登録パタンに定義されているマスク情報を取得します。連続領域バッファ *mask に各画素のマスクのオン／オフ情報が順に格納されます。
順序は左上の画素から右に進み、最後は右下の画素にくる順序
（Lib_srs_define_mask の図の左側に描いてある矢印の順序）です。
マスクがオンになっている画素には１が、オフになっている画素には０が与えれれます。

① ***ptn_name*** はマスク情報を取得したい登録パタンの名称です。

② ****mask*** はマスク情報を格納するバッファです。
　必要なサイズは（パタン画像の横サイズ）×（パタン画像の縦サイズ）です。

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。
-105	SRSERR_PATTERN	指定パタンが未登録のための異常終了です。
-107	SRSERR_MASK	指定パタンにはマスクが未定義です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○マスク情報を格納するバッファは呼出し側で確保してください。その画像の縦・横のサイズは Lib_srs_get_ptn_image_size で取得できます。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_get_speed

機　能

粗サーチのスピードタイプの取得

形　式

```
#include "f_srs.h"
int Lib_srs_get_speed( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

粗サーチのスピードタイプを取得します。
粗サーチは「通常処理」と「高速処理」の２段階の処理速度を選択できます。
「高速処理」ではサーチ画像を圧縮してサーチを実行するため、処理時間が「通常処理」と比較して約１／３から１／４になっています。

戻り値

値	定　　数	意　　味
0	SRS_NORMAL_SPEED	「通常処理」です。
1	SRS_HIGH_SPEED	「高速処理」です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○デフォルトは SRS_HIGH_SPEED、すなわち、「高速処理」です。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

Lib_srs_set_speed

機　能

粗サーチのスピードタイプの設定

形　式

```
#include "f_srs.h"
int Lib_srs_set_speed( int speed );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

粗サーチのスピードタイプを取得します。
粗サーチは「通常処理」と「高速処理」の２段階の処理速度を選択できます。
「高速処理」ではサーチ画像を圧縮してサーチを実行するため、処理時間が「通常処理」と比較して約１／３から１／４になっています。

① ***speed*** は設定するスピードのタイプです。

値	定　　数	意　　味
0	SRS_NORMAL_SPEED	「通常処理」
1	SRS_HIGH_SPEED	「高速処理」

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○デフォルトは SRS_HIGH_SPEED、すなわち、「高速処理」です。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

○Lib_srs_set_fine_srch_sw の最後【粗サーチと精サーチ】にあるように本サーチは粗サーチを実行した後、その結果を元に精サーチを行なっています。したがって、精サーチを実行する場合にはスピードタイプを「通常処理」にしても「高速処理」にしても精度には全く違いはありません。
以上より、精サーチを実行する場合には「高速処理」にすることをお勧めします。

Lib_srs_get_fine_srch_sw

機　能

精サーチ実行スイッチの取得

形　式

```
#include "f_srs.h"
int Lib_srs_get_fine_srch_sw( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

精サーチの実行スイッチのオン／オフの状態を取得します。

戻り値

値	定　　数	意　　味
0	SRS_FINE_SRCH_OFF	精サーチを実行しません。
1	SRS_FINE_SRCH_ON	精サーチを実行します。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○デフォルトは SRS_FINE_SRCH_ON、すなわち、精サーチを実行します。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

○Lib_srs_set_fine_srch_sw の最後【粗サーチと精サーチ】もご覧ください。

Lib_srs_set_fine_srch_sw

機　能

精サーチ実行スイッチの設定

形　式

```
#include "f_srs.h"
int Lib_srs_set_fine_srch_sw( int fine_sw );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
	○			○	

解　説

精サーチの実行スイッチを設定します。

① ***fine_sw*** は精サーチの実行スイッチです。

値	定　数	意　味
0	SRS_FINE_SRCH_OFF	精サーチを実行しない
1	SRS_FINE_SRCH_ON	精サーチを実行する

戻り値

値	定　数	意　味
0	NORMAL_RETURN	正常終了です。
-102	SRSERR_OPEN	Lib_srs_open が正常終了していないための異常終了です。

例

ファイル srs.c として csc902¥sample ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Ｓ回転サーチライブラリを使用する場合にはオープン Lib_srs_open を最初にコールし、正常終了されなければなりません。

○デフォルトは SRS_FINE_SRCH_ON、すなわち、精サーチを実行します。

○Ｓ回転サーチの各ライブラリのコールのタイミングについては冒頭の【Ｓ回転サーチの処理手順】をご覧ください。

【粗サーチと精サーチについて】

本サーチはパタンの大体の位置を求める「粗サーチ」とその結果を元に高精度に位置を求める「精サーチ」の２段階に分かれています。

粗サーチではまず最初にサーチ画像から登録パタンによらない特徴量を抽出した後、各パタンのサーチを実行するため、１枚の画像から異種パタン・複数個のパタンをサーチする場合でも処理時間はそれ程増加しません。一方で、精サーチは粗サーチの結果として得られる１つ１つの回答に対して位置を高精度に求める計算を実行するため、その処理時間は回答数に比例してかかります。

弊社の実験では粗サーチの精度は精サーチの精度の約１／５～１／１０という結果が得られています。すなわち、精サーチで１／１０画素の精度がでる場合には１／２～１画素の精度がでています。

それ程精度にはこだわらない場合で処理速度をより高速にしたい場合や、異種パタン・複数パタンを同一画像から多数検出したい場合などには粗サーチのみで試行してみてください。

4．直線検出ハフ変換ライブラリ

画面内に１本の線が存在しているとき、「最小自乗法」を用いれば、その直線を検出するのは容易です。
しかし、複数の直線が存在しているときは、ことはそう簡単ではありません。
ハフ変換はこのようなときに有効な方法です。

ハフ変換は、直線検出に限らずパラメータで表現できる図形（例えば、円や楕円）を画像中から検出するための手法ですが、パラメータの数が多くなると、処理時間や必要なワークメモリが膨大なものになります。
そこで当社では、直線検出用のハフ変換のみライブラリとして用意しました。

直線検出用のハフ変換の基本原理を簡単に説明すると次のようになります。

予め、ρ－θパラメータ平面をワークメモリ上に確保し、初期化しておきます。
エッジ検出後の画像においてエッジ画素（０以外の値を有する画素）の座標値$\left(X_I,Y_i\right)$を下に示す式に代入してρ－θパラメータ平面上に軌跡を描きます。
パラメータ平面上のある点を通過する軌跡の本数である累積度数が多い点を順次求めることにより、複数の直線を検出します。

$$\rho = X_i \cdot \cos(\theta) + Y_i \cdot \sin(\theta) \qquad (i = 1 , 2 ,\ldots , n)$$

ハフ変換について詳しいことを知りたいときは以下の書籍をご覧ください。

１）高木幹雄監修、“画像解析ハンドブック”、東京大学出版会
２）森俊二・坂倉栂子共著、“画像認識の基礎〔Ⅱ〕”、オーム社
３）長尾真著、“画像認識論”、コロナ社

Lib_lhough_open

機　能

直線検出ハフ変換のオープン

形　式

```
#include ″f_hough.h″
int  Lib_lhough_open( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

ワークメモリ上にハフ平面を生成し、ハフ変換に必要な初期化を行います。
ハフ平面に必要なワークメモリのサイズは以下の通りです。
処理ウインドウ始点（ｘｓ，ｙｓ），処理ウインドウ終点（ｘｅ，ｙｅ）で

$$W = (xe - xs + 1)/2$$

$$H = (ye - ys + 1)/2$$

とした時

ワークサイズ $= 720\sqrt{W^2 + H^2}$ （バイト）

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ありません	メモリ不足のためハフ平面が生成できません。
－２	ありません	２重にオープンしようとしました。

例　　カレントメモリ上の画像から直線を１０本抽出します。

```
#include  "f_hough.h"
#include  "f_graph.h"
#include  "f_video.h"

typedef unsigned char  uchar;
#define LINE_N         10

static void  draw_calc_line( double  a, double  b )
{
    double  d_x, d_y;
    int     x, y, fx, fy;

    fx = Lib_get_fx_size();
    fy = Lib_get_fy_size();

    if( - 1.0 <= a  &&  a <= 1.0 )
    {
        for( x = 0; x <= fx; x++ )
        {
            d_x = ( double )x;
            d_y = a * d_x + b;
            y = ( int )d_y;

            Lib_pset( x, y, GRAPH_DRAW );
        }
    }
    else
    {
        for( y = 0; y <= fy; y++ )
        {
            d_y = ( double )y;
            d_x = d_y / a - b / a;
            x = ( int )d_x;

            Lib_pset( x, y, GRAPH_DRAW );
        }
    }
}


void  main()
{
    static HLINE_T  line_coeff[LINE_N];
           uchar    *src;
           int      i, wa, wb, x, y, fx, fy;
           int      line_n, xs, ys, xe, ye;
           int      work_mem;
           double   m, b;

    if( ERROR_RETURN != ( work_mem = Lib_alloc_gray_memory() ) )
    {
        fx = Lib_get_fx_size();
        fy = Lib_get_fy_size();
```

```
        xs = ys = 0;
        xe = fx - 1;
        ye = fy - 1;
        Lib_set_stage_window( xs, ys, xe, ye );

        Lib_freeze( NOT_TRANSMIT );

        if( NORMAL_RETURN == Lib_any_cross( - 1, work_mem, 4, 128, 255 ) )
        {
            if( NORMAL_RETURN == Lib_lhough_open() )
            {
                src = ( uchar * )Lib_adrs_gray_memory( work_mem );

                for( y = ys + 1; y < ye; y++ )
                    for( x = xs + 1; x < xe; x++ )
                        if( 0 != *( src + y * fx + x ) )
                              Lib_lhough_voting( x, y );

                if( 0 <= ( line_n = Lib_lhough_detection
                                         ( LINE_N, 20, 10, line_coeff ) ) )
                {
                    for( i = 0; i < line_n; i++ )
                    {
                        if( line_coeff[i].b == 0 )
                        {
                            b = - ( double )line_coeff[i].c * 4096.0
                                                  / ( double )line_coeff[i].a;
                            wb = ( WORD )b;
                            Lib_drawline( wb, 0, wb, fy - 1 );
                        }
                        else
                        {
                            m = - ( double )line_coeff[i].a / ( double )
                                                              line_coeff[i].b;
                            b = - ( double )line_coeff[i].c * 4096.0
                                                  / ( double )line_coeff[i].b;
                            draw_calc_line( m, b );
                        }
                    }
                }
                else  Lib_printf( "Lib_lhough_detection error!¥n¥r" );

                if( ERROR_RETURN != Lib_lhough_close() )
                      Lib_printf( "Completed !¥n¥r" );
                else  Lib_printf( "Lib_lhough_close error!¥n¥r" );
            }
            else  Lib_printf( "Lib_lhough_open error !¥n¥r" );
        }
        else  Lib_printf( "Lib_any_cross error !¥n¥r" );
        Lib_free_gray_memory( work_mem );
    }
    else  Lib_printf( "Lib_alloc_gray_memory error !¥n¥r" );
}
```

留意事項

○ハフ平面を決定するために、カレントステージの処理範囲（ Lib_get_stage_window で得られる範囲）を使用しますので、本ライブラリ以前に当該範囲を設定しておいてください。

○本ライブラリを呼び出したら、最後に Lib_lhough_close を呼び出してください。
さもないと、ワークメモリの解放がなされません。

○同一ソフト内でノーマルモードと高解像度モードと切りかえて使用する場合、そのたびにｏｐｅｎし直す必要があります。

Lib_lhough_close

機　能

直線検出ハフ変換のクローズ

形　式

```
#include  ″f_hough.h″
int  Lib_lhough_close( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

ハフ平面を解放します。
これ以降、Lib_lhough_xxxx （xxxxは任意)は一切使えなくなります。
再び、ハフ変換を行うには、Lib_lhough_open を呼び出してください。

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	ハフ平面が存在しません。

例

Lib_lhough_open の例を参照してください。

留意事項

○本ライブラリの呼び出し前に必ず Lib_lhough_open を呼び出しておいてください。

Lib_lhough_voting

機　能

ハフ平面への投票

形　式

```
#include  ″f_hough.h″
int  Lib_lhough_voting( int  x, int  y );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

与えられた座標値をもとにハフ変換を行い、ハフ平面上に投票します。

① ***x*** は投票したいｘ座標値です。

② ***y*** は投票したいｙ座標値です。

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	ハフ平面が存在しません。

例

Lib_lhough_open の例を参照してください。

留意事項

○本ライブラリの呼び出し前に必ず Lib_lhough_open を呼び出しておいてください。

Lib_lhough_detection

機　能

ハフ変換による直線の検出

形　式

```
#include  "f_hough.h"
int  Lib_lhough_detection( int  line_n, int  region_r,
                                          int  region_q, HLINE_T  line_coeff[] );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

与えられた座標値から直線のハフ変換を行い、直線係数を求めます。

① ***line_n*** は検出したい直線の本数を１～１２８の範囲で指定します。

② ***region_r*** はハフ平面上の極大値決定のｒ軸方向領域サイズです。
　１以上の値を設定してください。実際のサイズはここで指定した値をもとに次のように計算されます。

$$r\text{軸の領域サイズ} = 2 * region_r + 1$$

③ ***region_q*** はハフ平面の極大値決定のθ軸方向領域サイズです。
　１以上の値を設定してください。実際のサイズはここで指定した値をもとに次のように計算されます。

$$\theta\text{軸の領域サイズ} = 2 * region_q + 1$$

④ ***line_coef[]*** 内には検出した直線の係数が格納されます。

構造体 HLINE_T は f_hough.h 内で次のように宣言されています。

```
typedef struct
  {
    int  a;      /* 係数ａを６５５３６倍した３２ビット整数値 */
    int  b;      /* 係数ｂを６５５３６倍した３２ビット整数値 */
    int  c;      /* 係数ｃを     １６倍した３２ビット整数値 */
  } HLINE_T;
```

戻り値

処理結果

値	定　　数	意　　味
０＜	ありません	実際に検出した本数です。(必ずしも、指定した本数line_nになるわけではありません)
－１	ERROR_RETURN	引き数が適当ではありません。

例

Lib_lhough_open の例を参照してください。

留意事項

○本ライブラリの呼び出し前に必ず Lib_lhough_open を呼び出し、なおかつ、Lib_lhough_voting も複数回呼び出しておいてください。

５．新直線検出ハフ変換ライブラリ

(エッジの向きを用いた直線検出ハフライブラリ)

直線検出ハフは直線が画像のどこにあるかわからない、しかも画像がノイジーである、といった場合でも直線を検出できる強力なツールとして知られています。しかし、例えば投票に時間がかかる、といった欠点もあります。

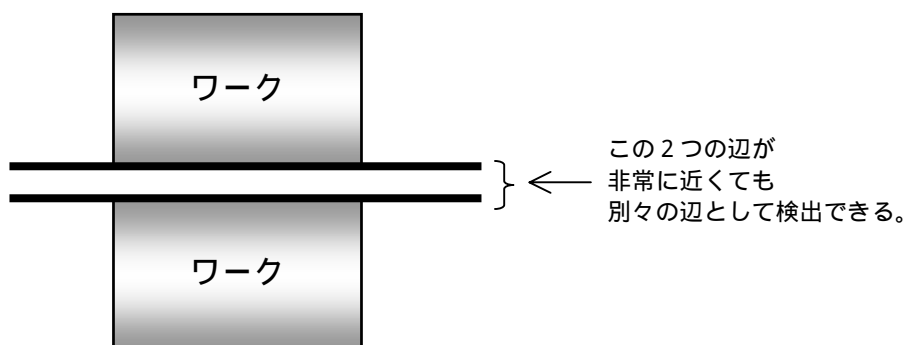
そこで、投票の対象となる点を「向きがわかっているエッジ点」とすることで、処理速度の高速化と検出能力の向上を計った新しい直線検出ハフをライブラリ化しました。

従来の直線検出ハフライブラリ(Lib_lhough_xxx)との主な相違点は以下の通りです。

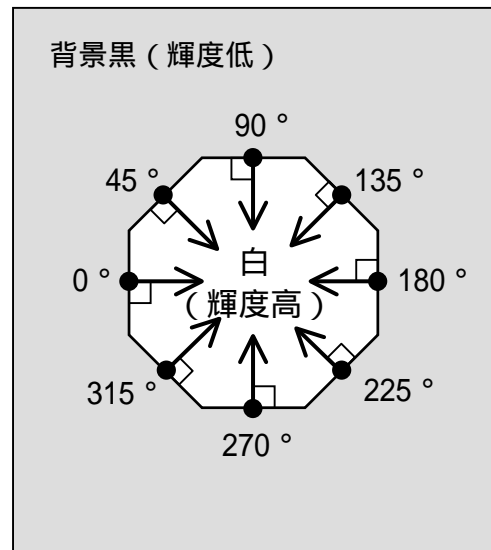
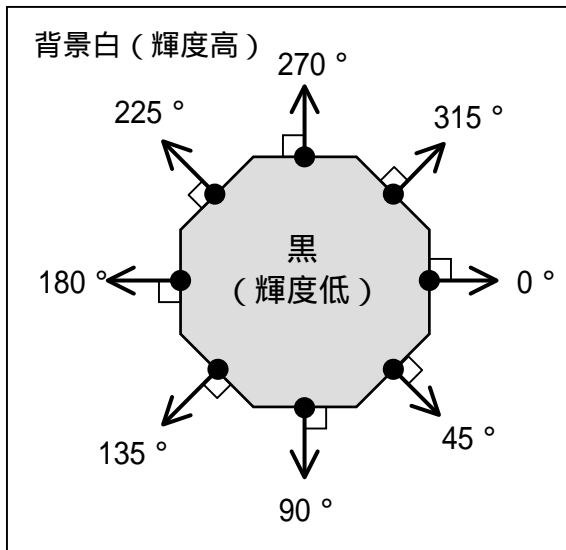
1. 欲しい直線には関係のない無駄な投票を減らすことで処理速度が大幅に短縮されました。同時に投票のピークが先鋭化されるため検出能力が向上しました。

2. 検出したい直線の傾きの範囲を指定できます。傾きは角度で表わされますが、このことについては次頁の **【エッジの向きと直線の傾きの関係について】** を参照してください。

3. 識別子により複数の直線検出ハフを切り分けられます。よって、例えば垂直線と水平線を独立に検出する、といったこともできます。

4. 例えば同色の長方形のワークが非常に接近して並置されている場合でもその辺を別々に検出することができます。

5. 従来の検出ライブラリ Lib_lhough_detection ではその処理時間が検出する直線の数に比例して増大していましたが、新しいライブラリでは検出直線の数が多くなってもそれほど処理時間は増大しません。

【エッジの向きと直線の傾きの関係について】

エッジの向きは画像の色で黒(輝度低)から白(輝度高)の方向を向いているものとします。
角度の単位は「度」とします。下の図を参考にしてください。

また、直線の傾き(向き)は角度で表わすことができますが、その角度はその直線上にある<u>エッジの角度</u>と同じものとします。

例えば、上左の図では、一番上の辺上にあるエッジの向きは２７０度ですから、この辺(直線)の傾きを表わす角度も２７０度となります。同様に考えて、一番左にある辺(直線)の傾きを表わす角度は１８０度、ということになります。

注意して頂きたい点は、直線の傾きを表わす角度の範囲は３６０度である、ということです。
上左の図の例では、一番下の辺の傾きは９０度であり、平行である一番上の辺とは１８０度差があることになります。これはエッジの向きが逆になっているからで、この性質を用いれば、上記の従来のハフとの相違点の４．が実現できることがわかります。

【注意】このライブラリはオープンするハフの数、投票範囲、検出する直線の向きに比例した、かなりの空きメモリを必要とします。したがって、これらの設定状況によっては４ＭＢのメモリを搭載した機種ではご使用になれない場合があります(メモリ不足によるエラーリターンになります)。このハフライブラリを有効にご使用になるために、１６ＭＢ以上のメモリを搭載した機種でのご使用を推奨します。

Lib_xlhough_open

機　能

新直線検出ハフのオープン

形　式

```
#include "f_hough.h"
void *Lib_xlhough_open( int st_q, int ed_q, int sx, int sy,
                                  int ex, int ey, int dummy );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

直線検出ハフの初期設定を指定し、必要なメモリを確保します。
直線検出ハフを行なう際、最初にコールし、正常終了しなければなりません。
さもなければ、次頁からの Lib_xlhough_xxx(xxx は任意)は一切使用できません。

①，② ***st_q***，***ed_q*** は検出したい直線の傾きの範囲を角度で表わしたものです
(冒頭の**【エッジの向きと直線の傾きの関係について】**を参照してください)。
　これらのパラメータは次式を満たすように指定してください。

$$-360 \leqq st_q \leqq 360,\quad 0 \leqq ed_q \leqq 360$$

$$st_q \leqq ed_q,\quad ed_q - st_q < 360$$

③～⑥ ***sx***，***sy***，***ex***，***ey*** は投票するエッジ点の存在する矩形領域を指定するパラメータです。(sx, sy) が左上の点、(ex, ey) が右下の点の座標を表わすように指定してください。

⑦ ***dummy*** は現バージョンではダミー変数としています。
　１を代入してください。

戻り値

値	定　数	意　味
0	NULL	引数が不適切かメモリ不足のための異常終了です。 この場合、投票、検出などの処理は実行できません。
０以外	ありません	正常終了で、返値は識別子です。

例

ファイル xlhough.c として csc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○この新直線ハフライブラリでは複数のハフを独立に扱うことができます。
例えば、水平線専用のハフとして、st_q = 85, ed_q = 95 でハフをオープンし、さらに、垂直線専用のハフとして、st_q = -5, ed_q = 5 で第２のハフをオープンすることができます。この際、Lib_xlhough_open の返値として得られる２つの識別子は異なっており、これにより、２つのハフを区別することができ、全く独立に扱うことができます。

○複数のハフをオープンする際に、オープンに要する処理時間は１回目と２回目以降では異なります。これは１回目では内部で共通に参照するテーブルを作成するためです。

○必要なワークメモリのサイズも１回目のオープン時と２回目以降のオープン時とでは異なります。
必要なワークメモリサイズはおおよそ以下の通りです；

$$w = (ex - sx + 1) / 2$$

$$h = (ey - sy + 1) / 2$$

$$q = (ed_q - st_q + 1)$$

とした時、

１回目のオープンで必要なワークメモリサイズ

$$= \left(2\sqrt{w^2 + h^2} + 1 \right) \times q \times 2Byte + \text{約}512\mathrm{K}Byte$$

２回目以降のオープンで必要なワークメモリサイズ

$$= \left(2\sqrt{w^2 + h^2} + 1 \right) \times q \times 2Byte$$

○本ライブラリをコールしたら、最後にクローズ Lib_xlhough_close を必ずコールしてください。
さもないと、本ライブラリで確保したワークメモリが解放されません。
本ライブラリをコールした回数と、Lib_xlhough_close をコールした回数とは最終的に一致していなければなりません。

Lib_xlhough_close

機　能

新直線検出ハフのクローズ

形　式

```
#include "f_hough.h"
int  Lib_xlhough_close( void *dscrp );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

Lib_xlhough_open で確保したワークメモリを解放します。

① ***dscrp*** はオープンで得られた識別子です。この識別子を持つハフがクローズされ、以降、Lib_xlhough_xxx(xxx は任意)は一切使用できなくなります。

戻り値

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了です。
－１	ERROR_RETURN	オープンが完了していないための異常終了です。

例

ファイル xlhough.c として csc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○あらかじめオープンが Lib_xlhough_open がコールされ、正常終了されてなければなりません。

○オープン Lib_xlhough_open をコールしたら、最後に本ライブラリを必ずコールしてください。
さもないと、本ライブラリで確保したワークメモリが解放されません。
Lib_xlhough_open をコールした回数と、本ライブラリをコールした回数とは最終的に一致していなければなりません。

Lib_xlhough_init_hough_sp

機　能

新直線検出ハフのハフ空間の初期化

形　式

```
#include "f_hough.h"
int Lib_xlhough_init_hough_sp( void *dscrp );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

ハフ投票空間を０に初期化します。

① ***dscrp*** はオープンで得られた識別子です。
　この識別子を持つハフ空間が初期化されます。

戻り値

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了です。
－１	ERROR_RETURN	オープンが完了していないための異常終了です。

例

ファイル xlhough.c として csc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○従来のハフのライブラリではオープン Lib_lhough_open か
　検出 Lib_lhough_detection がコールされたときにハフ空間が初期化されていましたが、新しいハフでは初期化は本ライブラリがコールされた場合にのみ実行されますので、ご留意ください。

○あらかじめオープンが Lib_xlhough_open がコールされ、正常終了されてなければなりません。

Lib_xlhough_edge_open

機　能

新直線検出ハフの方向付きエッジ配列のオープン

形　式

```
#include "f_hough.h"
int Lib_xlhough_edge_open( int mem_no, int thres,
                           int sx, int sy, int ex, int ey, QEDGE_T **edge );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

新直線検出ハフで投票される向き付きのエッジを配列で取得するためのライブラリです。
配列バッファは内部で必要なだけ確保されますので、呼出し側で確保する必要はありません。
エッジは mem_no で指定される濃淡画像の輝度を６ビットに圧縮した後、ソーベルフィルタを作用させることによって取得されます。
各エッジの向きは－１７９度から１８０度の整数で与えられます。

① ***mem_no*** はエッジを取得する対象の濃淡画像のメモリ番号です。

② ***thres*** はエッジ点と判定するためのしきい値です。
　thres = -1 とするとしきい値が自動決定され、エッジ点が抽出されます。
　1 ≦ thres ≦ 255 とすると、この値をしきい値としてエッジ点が抽出されます。
　この場合は上記の６ビット画像に濃淡画像ライブラリ Lib_sobel を XY_DIRECTION で作用させた場合の各画素値にたいして、このしきい値以上の点をエッジ点と判定します。

③～⑥ ***sx***, ***sy***, ***ex***, ***ey*** は投票するエッジ点を取得する矩形領域を指定するパラメータです。(sx, sy) が左上の点、(ex, ey) が右下の点の座標を表わすように指定してください。

⑦ *****edge*** は取得されたエッジ点の配列の先頭アドレスへのポインタです。
　構造体 QEDGE_T は f_hough.h の中で次のように定義されています。

```
typedef struct
{
    short   x;          /*  エッジのｘ座標  */
    short   y;          /*  エッジのｙ座標  */
    short   q;          /*  エッジの向き    */
}QEDGE_T;
```

エッジの向き q は整数で、-179 ～ 180 度の範囲で与えられます。

戻り値

値	定　　数	意　　味
0 <=	ありません	取得できたエッジ点の数です(０の場合、エッジ点がなかった、ということです)。
－１	ERROR_RETURN	オープンが完了していないか、引数が不適切か、またはメモリ不足のための異常終了です。

例

ファイル xlhough.c として csc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○しきい値 thres の決定のためのツールとして、Lib_xlhough_thres_test が用意されていますので、ご活用ください。

○本ライブラリは特定のハフに依存しないものなので、
オープン Lib_xlhough_open で与えられる識別子を引数に持っていません。
しかし、少なくとも１つのハフがオープンされていなければ使用できません。

○本ライブラリで取得したエッジが不必要になったら、
Lib_xlhough_edge_close にてエッジをクローズしてください。
さもないと、メモリが解放されません。
本ライブラリをコールした回数と、 Lib_xlhough_edge_close をコールした回数とは最終的に一致していなければなりません。

○本ライブラリでのエッジの取得範囲とハフのオープン Lib_xlhough_open で指定する矩形領域とはまったく関係がありません。ただし、実際に投票されるエッジ点はオープンで指定した領域内にある点でなければなりません。

Lib_xlhough_edge_close

機　能

新直線検出ハフの方向付きエッジ配列のクローズ

形　式

```
#include "f_hough.h"
void Lib_xlhough_edge_close( QEDGE_T *edge );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

Lib_xlhough_edge_open で取得したエッジを格納していた配列を解放します。

① ***edge** は Lib_xlhough_edge_open で取得したエッジの配列の先頭アドレスです。
　必ず Lib_xlhough_edge_open で得られたものと同じアドレスを入力してください。

戻り値

ありません。

例

ファイル xlhough.c として csc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○Lib_xlhough_edge_open で取得したエッジが不必要になったら、本ライブラリにてエッジをクローズしてください。さもないと、メモリが解放されません。
　Lib_xlhough_edge_open をコールした回数と、本ライブラリをコールした回数とは最終的に一致していなければなりません。

○本ライブラリは特定のハフに依存しないものなので、
　オープン Lib_xlhough_open で与えられる識別子を引数に持っていません。
　しかし、少なくとも１つのハフがオープンされていなければ使用できません。

Lib_xlhough_thres_test

機　能

エッジ取得の際のしきい値を決めるためのテスト

形　式

```
#include "f_hough.h"
int Lib_xlhough_thres_test( int org_mem_no, int bin_mem_no,
                            int test_thres, int sx, int sy, int ex, int ey );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

Lib_xlhough_edge_open でエッジを取得する際に使用するしきい値を決めるためのツールです。
test_thres で指定されるしきい値で取得されるエッジ点を２値画像に白で描画します。
エッジ点を画面に表示することでしきい値を決定しやすくするライブラリです。

① ***org_mem_no*** はエッジを取得する対象の濃淡画像のメモリ番号です。

② ***bin_mem_no*** はエッジ点を描画する２値画像メモリ番号です。
この２値画像は呼出し側で確保してください。

③ ***test_thres*** は試すしきい値です。このしきい値の意味については
Lib_xlhough_edge_open の解説を参照してください。

④～⑦ ***sx***, ***sy***, ***ex***, ***ey*** は投票するエッジ点を取得する矩形領域を指定するパラメータです。(sx, sy) が左上の点、(ex, ey) が右下の点の座標を表わすように指定してください。

戻り値

値	定　　数	意　　味
０<=	ありません	取得できたエッジ点の数です(０の場合、エッジ点がなかった、ということです)。
－１	ERROR_RETURN	オープンが完了していないか､引数が不適切か､またはメモリ不足のための異常終了です。

例

ファイル xlhough.c として csc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○本ライブラリは特定のハフに依存しないものなので、
オープン Lib_xlhough_open で与えられる識別子を引数に持っていません。
しかし、少なくとも１つのハフがオープンされていなければ使用できません。

○本ライブラリでのエッジの取得範囲とハフのオープン Lib_xlhough_open で指定する矩形領域とはまったく関係がありません。

○エッジ点を描画した２値画像を画像に表示するためには、基本ライブラリの
Lib_video_transmit, または Lib_xvideo_transmit が必要です。

Lib_xlhough_voting

機　能

新直線検出ハフのハフ空間への配列での投票

形　式

```
#include "f_hough.h"
int Lib_xlhough_voting( void *dscrp, QEDGE_T edge[], int edge_num, int vot_wid );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

向きを持つエッジ点群が何らかの方法で得られている場合に、これらをハフ空間へ投票します。
例えば、Lib_xlhough_edge_open で取得したエッジ点列であればここでの投票に使用できます。
また、以下の②で規定されている形式に合っていれば、Lib_xlhough_edge_open で取得したエッジでなくても使用できます。

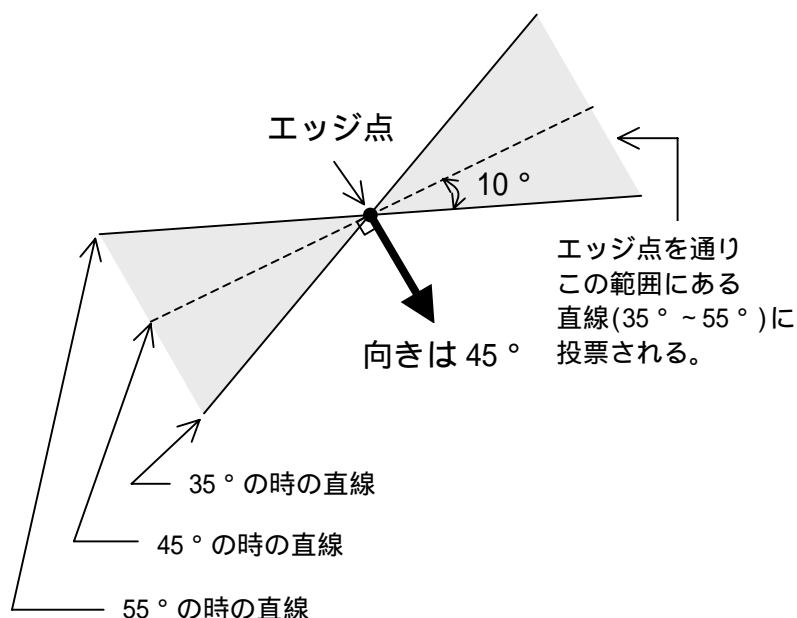
① ***dscrp** はオープンで得られた識別子です。
　この識別子を持つハフ空間に投票がされます。

② ***edge[]*** は投票するエッジ点です。構造体 QEDGE_T は f_hough.h の中で次のように定義されています。

```
typedef struct
{
    short   x;          /*  エッジのｘ座標  */
    short   y;          /*  エッジのｙ座標  */
    short   q;          /*  エッジの向き    */
}QEDGE_T;
```

エッジの向き q は整数で、-179 ～ 180 度の範囲になければなりません。

③ ***edge_num*** は②の配列 edge[] に格納されているエッジ点の数です。

④ ***vot_wid*** はハフ空間における投票の角度の範囲の片幅です。
例えば、vot_wid = 10 とした場合、あるエッジの向きが 45 度であれば、このエッジの投票される範囲は 35 度から 55 度となります(下図を参照してください)。

この範囲を小さくすれば投票にかかる処理時間も短縮され、また、投票結果のピークの先鋭化もはかれます。逆にエッジの向きがさほど信用出来ないような状況では真の回答の直線に投票されずに、検出に失敗する恐れもあります。
通常の画像に対しては vot_wid = 10 くらいから試してみることをお勧めします。

戻り値

値	定　　数	意　　味
0<=	ありません	投票に使用されたエッジ点の数です(０の場合、どの点も投票されなかった、ということです)。
－１	ERROR_RETURN	オープンが完了していないか､引数が不適切か､またはメモリ不足のための異常終了です。

例

ファイル xlhough.c として csc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○あらかじめオープンが Lib_xlhough_open がコールされ、正常終了されてなければなりません。

○投票するエッジ点の位置(x, y)はオープン Lib_xlhough_open で指定した矩形領域に入っていなければなりません。
本ライブラリでは処理時間短縮のため、エッジ点がこの領域に入っているかどうかのチェックは行なっていません。万が一、処理範囲外の点が入力された場合はメモリを壊す可能性があるため、動作の正常性は保証できません。ご留意ください。

○オープンで指定した検出直線の傾きの範囲によっては、本ライブラリで入力されても投票に寄与しないエッジ点がある可能性があります。
例えば、オープンで指定した直線の範囲が -10 度から 10 度で、本ライブラリの vot_wid が 20 度である場合､向きが 180 度であるエッジ点は仮に投票されるとしても 160 度から 200 度の範囲であり､直線の範囲 -10 度から 10 度にはないので､投票には寄与しません(無視されます)。

○Lib_xlhough_init_hough_sp をコールしない限りはハフ投票空間は初期化されないので､本ライブラリを複数回コールして投票値を累積させることも可能です。

Lib_xlhough_detection

機　能

新直線検出ハフによる直線の検出

形　式

```
#include "f_hough.h"
int Lib_xlhough_detection( void *dscrp, int rq_line_num,
                          int rgn_r, int rgn_q, XHLINE_T lines[] );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

投票された結果をもとに直線を検出します。

① ***dscrp*** はオープンで得られた識別子です。
この識別子を持つハフ空間から直線を検出します。

② ***rq_line_num*** は検出したい直線の数です。

③，④ ***rgn_r, rgn_q*** は似たような直線を検出しないようにするためのパラメータです。
このパラメータに関しては下記の留意事項の最後の説明**【引数 rgn_r，rgn_q について】**を参照してください。

⑤ ***lines[]*** は回答である直線の情報を格納する配列です。
この配列は呼出し側で事前に確保しておいてください。
配列のサイズは rq_line_num × sizeof(XHLINE_T) byte 必要です。
構造体 XHLINE_T は f_hough.h の中で、次のように定義されています。

```
typedef struct
{
    double  a;     /*                                           */
    double  b;     /*      直線の方程式 ax + by + c = 0 の係数   */
    double  c;     /*                                           */
    double  q;     /*  直線の傾きを表わす角度
                          (この直線上に乗るエッジの向きに一致)  */
    int     score; /*  投票数                                   */
}XHLINE_T;
```

(構造体のメンバ q については冒頭の**【エッジの向きと直線の傾きの関係について】**を参照してください)

戻り値

値	定　　数	意　　味
0<=	ありません	実際に検出された直線の本数です(０の場合、１本も直線が検出されなかった、ということです)。
－１	ERROR_RETURN	オープンが完了していないか、引数が不適切か、またはメモリ不足のための異常終了です。

例

ファイル xlhough.c として csc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○あらかじめオープンが Lib_xlhough_open がコールされ、正常終了されてなければなりません。
さらに、Lib_xlhough_voting によって複数の点がハフ空間に投票されていなければなりません。

○従来の直線検出ハフとは異なり、本ライブラリをコールしてもハフ空間は初期化されません。
ハフ空間を初期化するためには Lib_xlhough_init_hough_sp をコールする必要があります。

○ハフ空間の状況によっては要求した直線より少ない直線しか検出されないこともあります。

○従来の直線検出ハフとは直線を格納する構造体が異なります。
特に、直線の係数は浮動小数点で表現され、各係数の倍率は全て等倍です。
ご留意ください。

○【引数 rgn_r, rgn_q について】

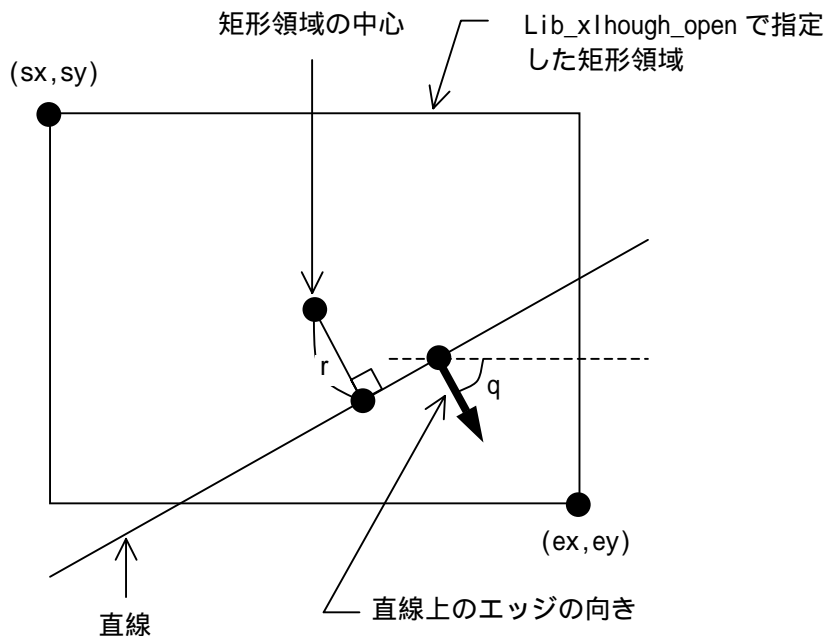
本ライブラリの内部では直線を

・(オープンで指定した)矩形領域の中心からの距離(単位は画素)

一般にｒとします

・傾き(単位は角度を表わす「度」)

一般にｑとします

で表わしています(下図を参照してください)。

似たような２つの直線とはｒとｑとがともに近い値を持つ、と考えられます。

そこで、似たような直線を検出しないようにするために、rgn_r, rgn_q を設定します。

この２つの値は

直線１のｒ、ｑの値がそれぞれ ｒ１、ｑ１

直線２のｒ、ｑの値がそれぞれ ｒ２、ｑ２

であるとき、

ｒ１、ｒ２の差の絶対値が rgn_r 以内で、しかも、ｑ１、ｑ２の差の絶対値が rgn_q 以内ならば、直線１、２のうち、投票数の多いどちらか一方だけ回答とせよ

というように使用されます。

したがって、

rgn_r を小さくすると、近くにある直線が複数検出される

rgn_q を小さくすると、傾きが似たような直線が複数検出される

ことになります。しかし、rgn_r, rgn_q をあまりに大きくすると別の複数の直線として検出したいものがどれか１本のみしか検出されない、ということになります。

どのような値を与えたらよいかは検出の対象となる画像中の直線の分布状況に依りますが、似たような直線が多数存在するような状況でなければ、

rgn_r、rgn_q ともに 5 〜 30 くらい

と設定して試してみることをお勧めします。

Lib_xlhough_refine_line

機　能

検出された直線を最小自乗法で求め直す

形　式

```
#include "f_hough.h"
int Lib_xlhough_refine_line( XHLINE_T *line, QEDGE_T edge[],
                                        int edge_num, int err_r, int err_q );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

指定された１本の直線とエッジ点群から、その直線の近くにある、と判定される点群を選び出し、これらに対して最小自乗法で直線の方程式を求め直します。

① ***line** は求め直したい直線です。Lib_xlhough_detection で得られた回答直線のうちの１本を選んで入力してください。本ライブラリが正常終了した場合、最小自乗法による結果で *line の内容は上書きされます。

構造体 XHLINE_T は f_hough.h の中で、次のように定義されています。

```
typedef struct
{
    double  a;     /*                                        */
    double  b;     /*      直線の方程式 ax + by + c = 0 の係数  */
    double  c;     /*                                        */
    double  q;     /*  直線の傾きを表わす角度
                          (この直線上に乗るエッジの向きに一致)  */
    int     score; /*  投票数                                 */
}XHLINE_T;
```

(構造体のメンバ q については冒頭の**【エッジの向きと直線の傾きの関係について】**を参照してください)

② **edge[]** は①の直線 *line の近くにある可能性のあるエッジ点群です。
Lib_xlhough_edge_open で取得したエッジ点群をそのまま入力しても結構ですし、何らかの処理によって、候補を絞ったものを入力しても結構です。
このエッジ点群の中から直線上にある点が選ばれます。

構造体 QEDGE_T は f_hough.h の中で次のように定義されています。

```
typedef struct
{
    short  x;          /*  エッジのｘ座標  */
    short  y;          /*  エッジのｙ座標  */
    short  q;          /*  エッジの向き    */
}QEDGE_T;
```

エッジの向き q は整数で、-179 ～ 180 度の範囲になければなりません。

③ ***edge_num*** は②の配列 edge[] に格納されているエッジ点の数です。

④ ***err_r*** はエッジ点が直線の近くにあるかどうかを判定するための値です。
具体的には、直線とエッジ点との距離が err_r 以内の場合、このエッジ点が直線の近くにある、と判定され最小自乗法の計算に使用されます(下図を参照してください)。
ただし、エッジ点が近くにあるかどうかの判定には次の⑤も使用されます。

⑤ ***err_q*** はエッジ点が直線の真のエッジ点であるかどうかを判定するための値です。
具体的には、直線の向きとエッジ点の向きとの差の絶対値が err_q 以内の場合、このエッジ点が直線の真のエッジである、と判定され最小自乗法の計算に使用されます(下図を参照してください)。
ただし、エッジ点が近くにあるかどうかの判定には上の④も使用されます。

例として err_q=10°とします。

エッジ点①は直線との距離が err_r 以内にあり、向きの差も err_q(=10°)以内なので直線の近くにある真のエッジ点として選ばれます。

エッジ点②は直線との距離が err_r 以内にあるが、向きの差も err_q よりも大きいので直線の近くにあるエッジ点には選ばれません。

エッジ点③,④は直線との距離が err_r より大きいので直線の近くのエッジ点には選ばれません。

戻り値

値	定　　数	意　　　味
正の値	ありません	正常終了で､返値は直線の近くにあるエッジ点の数です。
－１	ERROR_RETURN	メモリ不足か引数が不適切なための異常終了です。
－２	XLH_CALC_IMPOSSIBLE	直線の方程式の係数が不定になるような座標データが与えられたための異常終了です。
－３	XLH_CALC_OVERFLOWED	途中でオーバーフローを起こしたための異常終了です。

例

ファイル xlhough.c として csc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。

留意事項

○本ライブラリは特定のハフに依存しないものなので、
　オープン Lib_xlhough_open で与えられる識別子を引数に持っていません。

○従来の直線検出ハフや最小自乗法 Lib_calcline で使用されている直線の構造体とは異なります。
　特に、直線の係数は浮動小数点で表現され、各係数の倍率は全て等倍です。ご留意ください。

Lib_xlhough_support_open

機　能

検出された直線の付近にあるエッジ点群を求める(オープン)

形　式

```
#include "f_hough.h"
int Lib_xlhough_support_open( XHLINE_T *line, QEDGE_T edge[], int edge_num,
                              QEDGE_T **sprt, int *sort_type, int err_r, int err_q );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

指定された１本の直線とエッジ点群から、その直線の近くにある、と判定される点群を選び出します。得られる点群はその直線の傾きに応じて、ｘ座標の小さい順、またはｙ座標の小さい順にソートされています。どちらのソートが実行されるかは呼び出し側で指定することはできません。

① ***line** は指定する直線です。

構造体 XHLINE_T は f_hough.h の中で、次のように定義されています。

```
typedef struct
{
    double  a;     /*                                   */
    double  b;     /*  直線の方程式 ax + by + c = 0 の係数  */
    double  c;     /*                                   */
    double  q;     /*  直線の傾きを表わす角度
                        (この直線上に乗るエッジの向きに一致)  */
    int     score; /*  投票数                            */
}XHLINE_T;
```

(構造体のメンバ q については冒頭の**【エッジの向きと直線の傾きの関係について】**を参照してください)

② ***edge[]*** は①の直線 *line の近くにある可能性のあるエッジ点群です。
Lib_xlhough_edge_open で取得したエッジ点群をそのまま入力しても結構ですし、何らかの処理によって、候補を絞ったものを入力しても結構です。
このエッジ点群の中から直線上にある点が選ばれます。

構造体 QEDGE_T は f_hough.h の中で次のように定義されています。

```
typedef struct
{
    short   x;     /*  エッジのｘ座標   */
    short   y;     /*  エッジのｙ座標   */
    short   q;     /*  エッジの向き     */
}QEDGE_T;
```

エッジの向き q は整数で、-179 ～ 180 度の範囲になければなりません。

③ ***edge_num*** は②の配列 edge[] に格納されているエッジ点の数です。

④ ***sprt** は回答のエッジ点群を格納した配列の先頭アドレスへのポインタです。
この配列バッファはライブラリ内部で確保されています。
この配列バッファを開放するためには Lib_xlhough_support_close を使用します。

⑤ ****sort_type*** は回答のエッジ点群がソートされた方法です。
*sort_type = 1 の場合、回答エッジ点群はｘ座標の小さい順に並んでいます。
*sort_type = 2 の場合、回答エッジ点群はｙ座標の小さい順に並んでいます。

⑥ ***err_r*** はエッジ点が直線の近くにあるかどうかを判定するための値です。
意味は Lib_xlhough_refine_line にあるものと同じですので、そちらの説明をご覧ください。

⑦ ***err_q*** はエッジ点が直線の真のエッジ点であるかどうかを判定するための値です。
意味は Lib_xlhough_refine_line にあるものと同じですので、そちらの説明をご覧ください。

戻り値

値	定　数	意　味
正の値	ありません	正常終了で、返値は直線の近くにあるエッジ点の数です
－１	ERROR_RETURN	メモリ不足か引数が不適切なための異常終了です。

例

Lib_xlhough_detection で求めたある直線の近くにある点を画面に表示します。

```
#include "f_graph.h"
#include "f_hough.h"

#define  ERR_R  5
#define  ERR_Q  5

int  disp_support(
XHLINE_T  *line,          /*  直線                    */
QEDGE_T    edge[],        /*  投票に使用したエッジ配列  */
int        edge_num,      /*  そのエッジ数             */  )
{
    QEDGE_T  *sprt;                 /*  直線の近くの点     */
    int       sprt_num;             /*  直線の近くの点の数 */
    int       sort_type;
    int       i;


    /*  直線の近くのエッジ点のオープン  */
    sprt_num = Lib_xlhough_support_open( line, edge,
             edge_num, &sprt,  &sort_type, ERR_R, ERR_Q );

    if( 1 <= sprt_num )
    {
        for( i = 0 ; i < sprt_num ; i ++ )
        {
                /*  点の描画  */
                Lib_pset( sprt[ i ].x, sprt[ i ].y, GRAPH_DRAW );
        }

        /*  直線の近くのエッジ点のクローズ  */
        Lib_xlhough_support_close( sprt );
    }

    /*  返値は直線近くの点の数  */
    return( sprt_num );
}
```

留意事項

○本ライブラリは特定のハフに依存しないものなので、オープン Lib_xlhough_open で与えられる識別子を引数に持っていません。

○本ライブラリで取得したエッジが不必要になったら、Lib_xlhough_support_close にてエッジをクローズしてください。さもないと、メモリが解放されません。本ライブラリをコールした回数と、Lib_xlhough_support_close をコールした回数とは最終的に一致していなければなりません。

○配列 *sprt は 返値が１以上の場合のみに内部で確保されています。

○引数⑥、⑦の意味は Lib_xlhough_refine_line にあるものと同じですので、そちらの説明をご覧ください。

Lib_xlhough_support_close

機　能

直線付近のエッジ点群配列のクローズ

形　式

```
#include "f_hough.h"
void Lib_xlhough_support_close( QEDGE_T *sprt );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

Lib_xlhough_support_open で取得したエッジを格納していた配列を解放します。

① ***sprt*** は Lib_xlhough_support_open で取得したエッジの配列の先頭アドレスです。
必ず Lib_xlhough_support_open で得られたものと同じアドレスを入力してください。

戻り値

ありません。

例

Lib_xlhough_support_open の例をご覧ください。

留意事項

○Lib_xlhough_support_open で取得したエッジが不必要になったら、本ライブラリにてエッジをクローズしてください。さもないと、メモリが解放されません。
Lib_xlhough_support_open をコールした回数と、本ライブラリをコールした回数とは最終的に一致していなければなりません。

○本ライブラリは特定のハフに依存しないものなので、オープン Lib_xlhough_open で与えられる識別子を引数に持っていません。しかし、少なくとも１つのハフがオープンされていなければ使用できません。

6．エッジサーチライブラリ

エッジサーチは、等倍で回転なし一般化ハフ変換を応用したサーチアルゴリズムを用いています。

エッジサーチを行うときの処理の流れは次のとおりです。

パタンの登録

登録１）エッジ画像からエッジの重心を求めます。

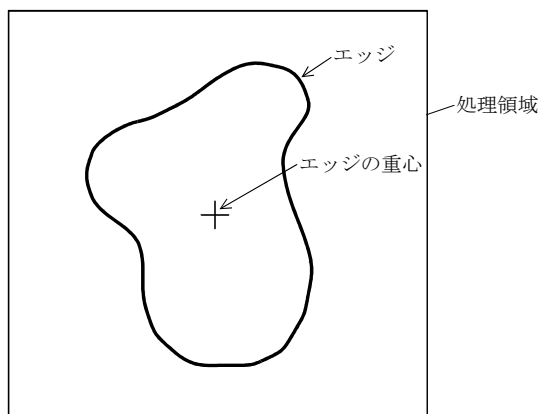



図１　エッジの重心

登録２）全エッジの中から、エッジを複数個選びだします。重心へ向かうベクトルを考え、このベクトルをパタンの特徴量とします。（濃度情報は全く使われません）エッジの数をユーザ側から指定できますので、選びだされたエッジの数だけベクトルを生成するわけです。
ベクトルが多いと、位置精度・認識率は堅牢になりますが、処理時間は遅くなります。
また、少ないとその逆で、処理時間は速いのですが、認識率は低くなってしまいます。
その時の画質により調整しなければならないのですが、１００本程度から始めてみる事をお薦めします。なお、ここで言うベクトルの事を、エッジサーチではシフトベクトルといいます。

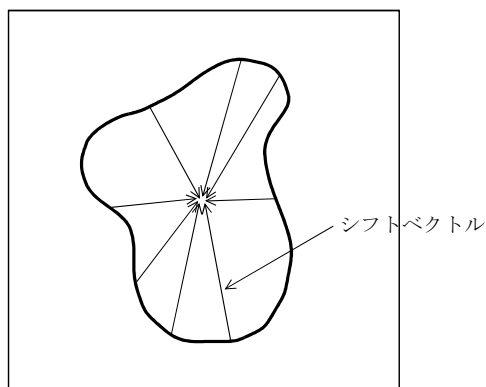



図２　シフトベクトル

パタンのサーチ

サーチ１）ハフ平面をワークメモリ上に確保し、その平面を０クリアしておきます。

サーチ２）サーチの処理範囲内にある入力エッジ画像を「登録２」で登録しておいたシフトベクトル分シフトします。

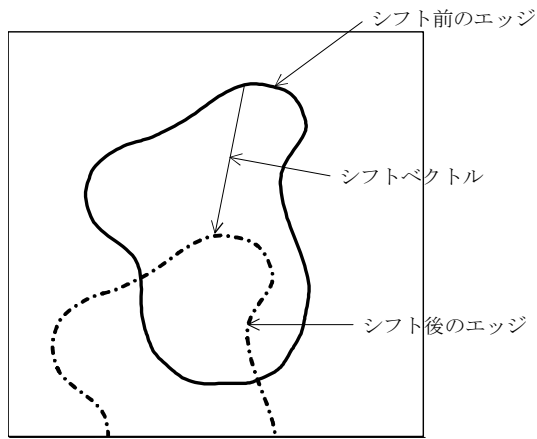



図３　エッジをシフト

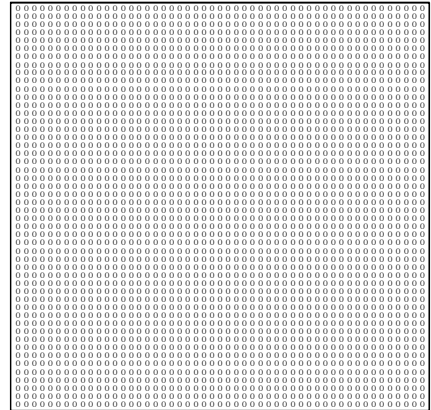



図４　ハフ平面（初期状態）

サーチ３）シフトされたエッジの座標に該当するハフ平面上の座標位置を＋１します。

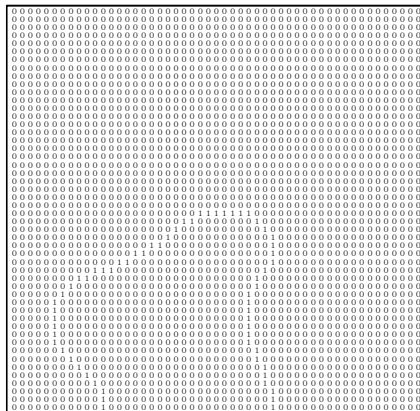



図５　ハフ平面（カウントアップ状態）

サーチ４）登録されているパタンのシフトベクトルの個数分、「サーチ２」および「サーチ３」を繰り返します。

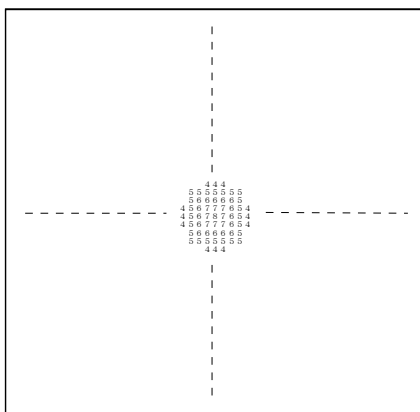



図６　ハフ平面（カウントアップ終了状態）

サーチ５）ハフ平面上で最大値を抽出します。

サーチ６）抽出された最大値およびその周辺の値から「２次曲面で最小２乗フィッティング」します。

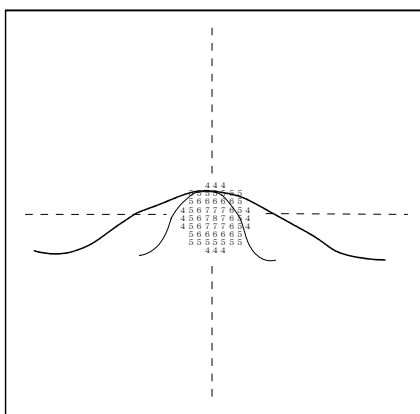



図７　ハフ平面（２次曲面フィッティング）

サーチ７）その位置を出力します。

ＣＳＣ９０Ｘシリーズでは、エッジサーチを行うプログラムが簡単に作成できるようライブラリを用意しました。
このライブラリを組み合わせることによってエッジサーチが実現できます。

Lib_es_init_dictionary

機　能

エッジサーチ用辞書（サーチパタン定義エリア）の初期化

形　式

```
#include ″f_edgsrc.h″
ESERR_T Lib_es_init_dictionary( void *dictionary, SIZE_T size );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジサーチに関する辞書（サーチパタン定義エリア）の初期化を行います。

① ***dictionary** は、辞書（サーチパタン定義エリア）の先頭番地です。本ライブラリでは、領域は確保しませんので、Lib_falloc や Lib_mlalloc等で、予め領域を確保して、その先頭番地を指定してください。

② ***size*** は、領域のバイト数です。確保してあるサイズより大きな値を指定すると、正常動作が保証されません。小さい分には何も問題ありません。

戻り値

処理結果

値	定　　数	意　　味
0	ESERR_NOTHING	正常終了しました。
－３	ESERR_NOT_ENOUGH_CAPACITY	指定されたサイズでは少なすぎます。
-101	ESERR_ARGUMENT1	１番目の引き数に誤りがあります。
-102	ESERR_ARGUMENT2	２番目の引き数に誤りがあります。

例

ファイルessmpl.cとしてcsc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。
「essmpl.c」はエッジサーチを体験するのに十分なプログラムとなっていますので、このプログラムを改造して使うのも１つの方法です。

留意事項

○全てのエッジサーチ用のライブラリに先立って本ライブラリをコールしてください。

○辞書のサイズは以下の計算式を参考にしてください。

$$dic_cap = dic_mng + (ptn_mng + 12 \times vect_n) \times ptn_n$$

記号の意味

dic_cap	：	辞書のサイズ（バイト）
dic_mng	：	辞書の管理領域（１６４バイト）
ptn_mng	：	サーチパタンの管理領域（１６０バイト）
$vect_n$	：	シフトベクトル数
ptn_n	：	サーチパタン数

例えば、シフトベクトルを１００本とすると
パタン数１０個で１３７６４バイト、１００個で１３６１６４バイトとなります。

Lib_es_set_max_edge

機　能

エッジサーチ用辞書（サーチパタン定義エリア）へ取り込み最大エッジ数の登録

形　式

#include　″f_edgsrc.h″

９０Ｘ

ESERR_T　Lib_es_set_max_edge(void　***dictionary**, unsigned long　***max_edge***);

FVL/LNX

ESERR_T　Lib_es_set_max_edge(void　***dictionary**, unsigned int　***max_edge***);

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジ画像から取り込んでくるエッジの最大数を設定します。

① ***dictionary*** は、辞書（サーチパタン定義エリア）の先頭番地です。
　本ライブラリをコールする前に、Lib_es_init_dictionary を呼び出しておく必要があります。

② ***max_edge*** はエッジ画像から切り出してくる最大エッジ数を指定します。
　ワークメモリが十分あるときは 50000程度、少ないときは、10000程度を目安にしてください。（１エッジにつき８バイト必要になります。）この値が小さいと、ノイズの多い画像には対応できなくなります。

戻り値

処理結果

値	定　　数	意　　味
0	ESERR_NOTHING	正常終了しました。
－１	ESERR_UNDEFINE_DICT	エッジサーチ用辞書が初期化されていません。
-101	ESERR_ARGUMENT1	１番目の引き数に誤りがあります。
-102	ESERR_ARGUMENT2	２番目の引き数に誤りがあります。

例

ファイルessmpl.cとしてcsc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。
「essmpl.c」はエッジサーチを体験するのに十分なプログラムとなっていますので、このプログラムを改造して使うのも１つの方法です。

留意事項

○本ライブラリの最大エッジ数はエッジサーチの処理時間には影響しません。
　メインメモリの容量に関係するのみですので、許す限り大きくしておいてください。

○ここで設定した max_edge の情報は、Lib_es_reg_pattern 及び Lib_es_calculation 内で使われます。

Lib_es_change_dictionary_size

機　能

エッジサーチ用辞書（サーチパタン定義エリア）のサイズ変更

形　式

#include ″f_edgsrc.h″
ESERR_T　Lib_es_change_dictionary_size(void　***dictionary**, SIZE_T　***set_size***);

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジサーチ用辞書（サーチパタン定義エリア）のサイズ変更をします。
Lib_falloc や Lib_mlalloc 等により領域を変更した際には、合わせて本ライブラリをコールしてください。

① ***dictionary*** は、辞書（サーチパタン定義エリア）の先頭番地です。
本ライブラリをコールする前に、Lib_es_init_dictionary をコールしておく必要があります。

② ***set_size*** は、変更後のサイズ（バイト数）です。
現状のサイズに対する追加分や削減分ではなく、直接希望のサイズを指定してください。
既にパタンが登録されている場合、そのサイズ（辞書の残サイズ）より小さくする事はできません。

戻り値

処理結果

値	定　数	意　味
0	ESERR_NOTHING	正常終了しました。
－１	ESERR_UNDEFINE_DICT	エッジサーチ用辞書が初期化されていません。
-101	ESERR_ARGUMENT1	１番目の引き数に誤りがあります。
-102	ESERR_ARGUMENT2	２番目の引き数に誤りがあります。

例

辞書サイズを set_size に変更します。

```
#include  "f_edgsrc.h"

int  change_process(
void    *dict_area;    /* 入力：辞書            */
SIZE_T  set_size;      /* 入力：変更したいサイズ    */ )
{
      SIZE_T min_size;    /*最小辞書サイズ*/
      SIZE_T max_size;    /*最大辞書サイズ*/
      SIZE_T remnant_size;/*既に設定してある辞書の残サイズ*/
      SIZE_T whole_size;  /*既に設定してある辞書の全サイズ*/
      SIZE_T usable_size; /*既に設定してある辞書の
                                サーチパタンの登録可能な全サイズ*/
      ESERR_T status;     /*エラー報告*/

      if( ESERR_NOTHING != ( status = Lib_es_get_dictionary_size
               ( dict_area, &whole_size, &usable_size, &remnant_size ) ) )
      {
          min_size = whole_size - remnant_size;
          max_size = MAX_DICT_SIZE;
          if( min_size < set_size  &&  set_size <= max_size )
              if( ESERR_NOTHING != ( status = Lib_es_change_dictionary_size
                                        ( dict_area, set_size ) ) )
                 Lib_es_error_message( status );
      }
      else  Lib_es_error_message( status );

     return( ( status == ESERR_NOTHING ) ? NORMAL_RETURN : ERROR_RETURN );
 }
```

留意事項

○ここでのサイズ変更はあくまでも辞書の管理上必要となるもので、辞書の領域を広げるものではありません。
実際のサイズ変更は本ライブラリをコールする前に、Lib_falloc や Lib_mlalloc 等により領域を変更しておいてください。

Lib_es_get_dictionary_size

機　能

エッジサーチ用辞書（サーチパタン定義エリア）のサイズ情報の取得

形　式

```
#include ″f_edgsrc.h″
ESERR_T  Lib_es_get_dictionary_size( void  *dictionary, SIZE_T  *whole_size,
                                     SIZE_T  *usable_size, SIZE_T  *remnant_size );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジサーチ用辞書（サーチパタン定義エリア）のサイズ情報を取得します。

① ***dictionary*** は、辞書（サーチパタン定義エリア）の先頭番地です。
本ライブラリをコールする前に、Lib_es_init_dictionary をコールしておく必要があります。

② ***whole_size*** は、辞書の全サイズ（バイト数）です。
Lib_es_init_dictionary や Lib_es_change_dictionary_size で設定したsize の値です。

③ ***usable_size*** は、サーチパタンが登録可能な全サイズ（バイト数）です。
*whole_size から辞書の管理領域（１６４バイト）を引いたものが本サイズになります。

④ ***remnant_size*** は、これからサーチパタンが登録可能なサイズ（バイト数）で、サーチパタンが登録できるか否かは、このサイズを確認することになります。

戻り値

処理結果

値	定　　数	意　　味
0	ESERR_NOTHING	正常終了しました。
－１	ESERR_UNDEFINE_DICT	エッジサーチ用辞書が初期化されていません。
-101	ESERR_ARGUMENT1	１番目の引き数に誤りがあります。
-102	ESERR_ARGUMENT2	２番目の引き数に誤りがあります。
-103	ESERR_ARGUMENT3	３番目の引き数に誤りがあります。
-104	ESERR_ARGUMENT4	４番目の引き数に誤りがあります。

例

Lib_es_change_dictionary_size の例を参照してください。

留意事項

○ *remnant_size がサーチパタンを登録するのに十分でないときは、領域を確保して、更に、Lib_es_change_dictionary_size をコールしてサイズを大きくしてください。

Lib_es_get_pattern_n

機　能

エッジサーチ用辞書（サーチパタン定義エリア）内のサーチパタン数取得

形　式

#include　″f_edgsrc.h″

９０X

ESERR_T　Lib_es_get_pattern_n(void　***dictionary**,　unsigned long　***pattern_n**　);

FVL/LNX

ESERR_T　Lib_es_get_pattern_n(void　***dictionary**,　unsigned int　***pattern_n**　);

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジサーチ用辞書（サーチパタン定義エリア）内に登録されているサーチパタン数を取得します。

① ***dictionary** は、辞書（サーチパタン定義エリア）の先頭番地です。
本ライブラリをコールする前に、Lib_es_init_dictionary をコールしておく必要があります。

② ***pattern_n** は、辞書に登録されているサーチパタン数です。

戻り値

処理結果

値	定　　数	意　　味
0	ESERR_NOTHING	正常終了しました。
－１	ESERR_UNDEFINE_DICT	エッジサーチ用辞書が初期化されていません。
-101	ESERR_ARGUMENT1	１番目の引き数に誤りがあります。
-102	ESERR_ARGUMENT2	２番目の引き数に誤りがあります。

例

ファイルessmpl.cとしてcsc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。
「essmpl.c」はエッジサーチを体験するのに十分なプログラムとなっていますので、このプログラムを改造して使うのも１つの方法です。

留意事項

ありません。

Lib_es_get_pattern_name

機　能

エッジサーチ用辞書（サーチパタン定義エリア）内のサーチパタン名取得

形　式

```
#include  "f_edgsrc.h"
ESERR_T  Lib_es_get_pattern_name( void  *dictionary, int page,
                                         unsigned char  *name );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジサーチ用辞書（サーチパタン定義エリア）内に登録されているサーチパタンの名称を取得します。

① ***dictionary*** は、辞書（サーチパタン定義エリア）の先頭番地です。
本ライブラリをコールする前に、Lib_es_init_dictionary をコールしておく必要があります。

② ***page*** は、辞書内のページです。ページは、登録の古い順に０から昇順になっています。
例えば、サーチパタンを「Ａ」「Ｂ」「Ｃ」という順に登録して、更に、「Ｂ」を変更した場合は、ページ０が「Ａ」、ページ１が「Ｃ」、ページ２が「Ｂ」になっています。

③ ***name*** はサーチパタン名称が格納されます。サーチパタン名がコピーされますので、コールする側で５文字以上の領域を確保しておく必要があります。
（コールする側でのプログラム例：　unsigned char　name[5]; ）

戻り値

処理結果

値	定　　数	意　　味
０	ESERR_NOTHING	正常終了しました。
－１	ESERR_UNDEFINE_DICT	エッジサーチ用辞書が初期化されていません。
-101	ESERR_ARGUMENT1	１番目の引き数に誤りがあります。
-102	ESERR_ARGUMENT2	２番目の引き数に誤りがあります。
-103	ESERR_ARGUMENT3	３番目の引き数に誤りがあります。

例

ファイルessmpl.cとしてcsc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。
「essmpl.c」はエッジサーチを体験するのに十分なプログラムとなっていますので、このプログラムを改造して使うのも１つの方法です。

留意事項

ありません。

Lib_es_reg_pattern

機　能

エッジサーチ用辞書（サーチパタン定義エリア）へサーチパタン登録

形　式

```
#include  ″f_edgsrc.h″
```

９０Ｘ

```
ESERR_T  Lib_es_reg_pattern( void  *dictionary, unsigned char  *name,
                             void  *edge, BOX_T  *pattern_area,
                                unsigned long  vector_n, PNT_T  *coord );
```

FVL/LNX

```
ESERR_T  Lib_es_reg_pattern( void  *dictionary, unsigned char  *name,
                             void  *edge, BOX_T  *pattern_area,
                                unsigned int  vector_n, PNT_T  *coord );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジサーチ用辞書（サーチパタン定義エリア）内へサーチパタンを登録します。

① ***dictionary*** は、辞書（サーチパタン定義エリア）の先頭番地です。
　本ライブラリをコールする前に、Lib_es_init_dictionary をコールしておく必要があります。

② ***name*** は、登録する際のサーチパタン名称です。
　１～４文字のＡＳＣＩＩ文字で指定してください。
　既に登録されているサーチパタン名を指定すると、その内容が更新されます。

③ ***edge*** はエッジが格納されているグレイメモリの先頭番地です。
　エッジでないものは「０」、エッジは「０以外」として扱いますので、そのようになるようにエッジ検出フィルタ等を調整してください。

④ ***pattern_area*** は *edge 中でのサーチパタンを切り出す領域です。
　この範囲内でシフトベクトル（これがサーチパタンになる）を抽出して登録します。

なお、BOX_T は以下のように定義されています。

```
typedef struct
{
    PNT_T  st;
    PNT_T  ed;
}   BOX_T;
```

上記中に出てくるPNT_Tは以下のように定義されています。

```
typedef struct
{
    int  x;
    int  y;
}   PNT_T;
```

⑤ ***vector_n*** はシフトベクトル数で、１～１０００の範囲で指定できます。
シフトベクトルを多くすると位置精度や認識率が向上しますが、処理時間が長くなります。複雑形状のサーチパタンで、１００程度、簡単なもので、３０程度を目安とするとよいでしょう。認識率が悪ければ、値をより大きく、処理速度が遅ければ、より少なくしてください。
なお、ver.4.3以降は０が指定できるようになりました。
０を指定すると自動的にシフトベクトル数を決定します。

⑥ ****coord*** は Lib_es_calculation で返ってくるサーチパタン上の座標位置です。
といっても、サーチパタン上にある必要はないので、妥当とおもわれる位置を入力してください。
Ｘ軸座標値・Ｙ軸座標値共に１０倍値を指定してください。
PNT_T は項④のように定義されています。
なお、ver.4.3以降は０が指定できるようになりました。
０を指定すると自動的にシフトベクトル数を決定します。

戻り値

処理結果

値	定　　数	意　　味
0	ESERR_NOTHING	正常終了しました。
－１	ESERR_UNDEFINE_DICT	エッジサーチ用辞書が初期化されていません。
－３	ESERR_NOT_ENOUGH_CAPACITY	辞書内の残容量が少なすぎて登録できません。
－４	ESERR_NOT_ENOUGH_MEMORY	ワークメモリが足りません。
－６	ESERR_NOT_FOUND_EDGE	エッジ画像内にエッジが見つかりません。
－７	ESERR_TOO_MANY_EDGE	エッジ画像内のエッジが多すぎます。
－９	ESERR_NOT_FOUND_VECTOR	良好なシフトベクトルが見つかりません。
-101	ESERR_ARGUMENT1	１番目の引き数に誤りがあります。
-102	ESERR_ARGUMENT2	２番目の引き数に誤りがあります。
-103	ESERR_ARGUMENT3	３番目の引き数に誤りがあります。
-104	ESERR_ARGUMENT4	４番目の引き数に誤りがあります。
-105	ESERR_ARGUMENT5	５番目の引き数に誤りがあります。
-106	ESERR_ARGUMENT6	６番目の引き数に誤りがあります。
-999	ESERR_INTERNAL_PROCESS	内部処理に障害が発生しました。 辞書の内容が破壊されている可能性があります。

例

ファイルessmpl.cとしてcsc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。
「essmpl.c」はエッジサーチを体験するのに十分なプログラムとなっていますので、このプログラムを改造して使うのも１つの方法です。

留意事項

○エッジ座標がきれいに出ていないと、位置精度・認識率共に低下しますので、なるべくノイズのないエッジ画像をサーチパタンにしてください。

○処理結果で ESERR_NOT_ENOUGH_MEMORY が返された時は、Lib_es_set_max_edge の値を小さくしてください。
また、ESERR_TOO_MANY_EDGE が返された時は、Lib_es_set_max_edgeのmax_edge の値を大きくしてください。

Lib_es_del_pattern

機　能

エッジサーチ用辞書（サーチパタン定義エリア）からサーチパタン消去

形　式

```
#include  "f_edgsrc.h"
ESERR_T  Lib_es_del_pattern( void  *dictionary, unsigned char  *name );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジサーチ用辞書（サーチパタン定義エリア）内に登録されているサーチパタンを消去します。

① ***dictionary*** は、辞書（サーチパタン定義エリア）の先頭番地です。
　本ライブラリをコールする前に、Lib_es_init_dictionary をコールしておく必要があります。

② ***name*** は、消去したいサーチパタン名です。
　１～４文字のＡＳＣＩＩ文字で指定してください。
　辞書内に登録されていない文字を指定すると、エラーになります。

戻り値

処理結果

値	定　　数	意　　味
０	ESERR_NOTHING	正常終了しました。
－１	ESERR_UNDEFINE_DICT	エッジサーチ用辞書が初期化されていません。
－５	ESERR_NOT_FOUND_PATTERN	辞書内に指定されたサーチパタンが見つかりませんでした。
-101	ESERR_ARGUMENT1	１番目の引き数に誤りがあります。
-102	ESERR_ARGUMENT2	２番目の引き数に誤りがあります。
-999	ESERR_INTERNAL_PROCESS	内部処理に障害が発生しました。辞書の内容が破壊されている可能性があります。

例

ファイルessmpl.cとしてcsc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。
「essmpl.c」はエッジサーチを体験するのに十分なプログラムとなっていますので、このプログラムを改造して使うのも１つの方法です。

留意事項

ありません。

Lib_es_calculation

機　能

エッジサーチ実行

形　式

```
#include "f_edgsrc.h"
ESERR_T  Lib_es_calculation( void  *dictionary, unsigned char  *name,
                                  void  *edge, BOX_T  *search_area,
                                   int  pick_n, int  rand_n, PNT_T  *position );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジサーチを実行します。

① ***dictionary** は、辞書（サーチパタン定義エリア）の先頭番地です。
本ライブラリをコールする前に、Lib_es_init_dictionary をコールしておく必要があります。

② ***name** は、サーチしたいサーチパタン名です。
１～４文字のＡＳＣＩＩ文字で指定してください。
辞書に登録されていない文字を指定すると、エラーになります。

③ ***edge** はエッジが格納されているグレイメモリの先頭番地です。
エッジでないものは「０」、エッジは「０以外」として扱いますので、そのようになるようにエッジ検出フィルタ等を調整してください。

④ ***search_area** は *edge 中でサーチしたい領域です。

なお、BOX_T は以下のように定義されています。

```
typedef struct
{
    PNT_T  st;
    PNT_T  ed;
}   BOX_T;
```

上記中に出てくるPNT_Tは以下のように定義されています。

```
typedef struct
{
    int  x;
    int  y;
}   PNT_T;
```

⑤ ***pick_n*** は抽出したい個数です。１～１００の範囲で指定できます。
サーチエリア内にここで指定する値以下の個数しかないときでも、強制的にこの数分だけサーチしますので、必要最低限の値にしておいてください。

⑥ ***rand_n*** はエッジ画像内のエッジに対して間引く係数です。１～１００の範囲で設定できます。
具体的には、３を指定したとすると、エッジ画像内の全エッジに対して １／３のエッジをランダムに選びだし、そのエッジのみからエッジサーチ計算をします。
このパラメータは計算時間を速くしたいときのみ２以上の数値を設定するようにして、通常は１を指定してください。位置精度に敏感に反応します。

⑦ ****position*** はサーチ結果座標で、１０倍値になります。
Lib_es_reg_pattern で指定した、*coord の座標位置に該当します。
コールする側で pick_n 分の容量を確保しておいてください。
（コールする側でのプログラム例： PNT_T position[pick_n]; ）
なお、PNT_T は項④のように定義されています。

戻り値

処理結果

値	定　数	意　味
０	ESERR_NOTHING	正常終了しました。
－１	ESERR_UNDEFINE_DICT	エッジサーチ用辞書が初期化されていません。
－４	ESERR_NOT_ENOUGH_MEMORY	ワークメモリが足りません。
－５	ESERR_NOT_FOUND_PATTERN	サーチしようとしているパタンが辞書内に見つかりません。
－６	ESERR_NOT_FOUND_EDGE	エッジ画像内にエッジが見つかりません。
－７	ESERR_TOO_MANY_EDGE	エッジ画像内のエッジが多すぎます。
－８	ESERR_NOT_CALCULATION	最小２乗計算に失敗しました。
-101	ESERR_ARGUMENT1	１番目の引き数に誤りがあります。
-102	ESERR_ARGUMENT2	２番目の引き数に誤りがあります。
-103	ESERR_ARGUMENT3	３番目の引き数に誤りがあります。
-104	ESERR_ARGUMENT4	４番目の引き数に誤りがあります。
-105	ESERR_ARGUMENT5	５番目の引き数に誤りがあります。
-106	ESERR_ARGUMENT6	６番目の引き数に誤りがあります。
-107	ESERR_ARGUMENT7	７番目の引き数に誤りがあります。
-999	ESERR_INTERNAL_PROCESS	内部処理に障害が発生しました。辞書の内容が破壊されている可能性があります。

例

ファイルessmpl.cとしてcsc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。
「essmpl.c」はエッジサーチを体験するのに十分なプログラムとなっていますので、このプログラムを改造して使うのも１つの方法です。

留意事項

○処理結果で ESERR_NOT_ENOUGH_MEMORY が返された時は、Lib_es_set_max_edge の値を小さくしてください。
また、ESERR_TOO_MANY_EDGE が返された時は、Lib_es_set_max_edge の max_edge を大きくしてください。

Lib_es_error_message

機　能

エッジサーチ用エラーメッセージ表示

形　式

```
#include ″f_edgsrc.h″
void  Lib_es_error_message( ESERR_T  err_code );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジサーチに関するライブラリ群から出力されるエラーをメッセージとしてモニタ上に表示します。

① ***err_code*** は、ESERR_T 内で定義されているエラーコードです。

戻り値

ありません。

例

ファイルessmpl.cとしてcsc90*¥sample(fv904¥sample)ディレクトリにインストールされていますので、そちらを参照してください。
「essmpl.c」はエッジサーチを体験するのに十分なプログラムとなっていますので、このプログラムを改造して使うのも１つの方法です。

留意事項

○エラー表示に本ライブラリを使わなければならないということはありません。

7．濃淡エッジ計測ライブラリ

このライブラリは、ＩＣのピン等の同じ形状で一列に並んでいる物を対象にしています。
エッジを計測して幅、ピッチ、中心座標を出力する関数とエッジ位置を出力する関数があります。
前者については以下に示す２種類の使い方があります。

（１）出力値を画素単位にしたい場合は座標変換係数を２５６としてエッジ測定を行い、出力値を浮動小数点型に変換後１６．０で割ってください。

（２）出力値を物理寸法等にしたい場合は、予めエッジ平均測定を行い、座標変換係数を求めた後、エッジ測定を実行してください。

エッジ計測プログラム例－１

変換係数を算出後、計測を行い、幅とピッチを出力します。

```
#include "f_em.h"      /* エッジ計測のヘッダーファイル */
#include "f_stdio.h"
#include "f_graph.h"
#include "f_video.h"

void main ( void )
{
    EM_AVR_INSPECTION avr;    /* エッジ平均測定の結果と変換係数を格納する構造体 */
    EM_INSPECTION inspection; /* エッジ測定結果を格納する構造体                */
    int i;
    int pitch = 1000;         /* ピッチ = 1000ミクロン  */
    int width = 0;            /* 幅 = ? */

    if ( Lib_em_inspection_open() == NORMAL_RETURN )/* エッジ計測の開始 */
    {
        Lib_drawline( 100, 100, 400, 100 );
        Lib_init_cursor();
        Lib_display_message( 400, 400, "開始", "係数算出" );

        Lib_freeze( TRANSMIT );
        Lib_em_avr_inspection( 100, 100, 400, 100, 50, 50,
                               1, 1000, 1000, &avr ); /* エッジ平均測定      */
        Lib_em_calib( &avr, pitch, width );            /* 変換係数を求める    */
        Lib_em_edge_disp();                            /* エッジ位置を表示する */
        printf( "¥r¥nfactor:%d", avr.factor );
        Lib_freerun();

        Lib_display_message( 400, 400, "開始", "エッジ計測" );

        Lib_cls( LINE_PLANE, 0 );
        Lib_drawline( 100, 100, 400, 100 );
        Lib_freeze( TRANSMIT );
        Lib_em_inspection( 100, 100, 400, 100, 50, 50,
                           1, 1000, 1000, avr.factor, &inspection ); /* エッジ測定 */
        printf( "¥r¥nnum:%d", inspection.num );
        for ( i = 0; i < inspection.num; i++ )
        {
            printf( "¥r¥nx:%d pitch:%d width:%d", inspection.x[i]/16,
                    inspection.pitch[i], inspection.width[i] );
        }
        Lib_em_edge_disp();          /* エッジ位置を表示する */
        Lib_em_inspection_close();   /* エッジ計測の終了     */
        Lib_freerun();

        Lib_display_message( 400, 400, "終了", "エッジ計測" );
    }
}
```

エッジ計測プログラム例－２

エッジ計測を行い、エッジ位置を出力します。

```
#include "f_em.h"      /* エッジ計測のヘッダーファイル */
#include "f_stdio.h"
#include "f_graph.h"
#include "f_video.h"

void main ( void )
{
    EM_EDGE_INFO edge_info;
    int i;
    char string[64];

    Lib_init_cursor();

    if ( Lib_em_inspection_open() == NORMAL_RETURN ) /* エッジ検査の開始 */
    {
        Lib_drawline( 100, 240, 400, 240 );
        Lib_display_message( 400, 400, "開始", "エッジ計測" );

        Lib_freeze( TRANSMIT );

        Lib_em_edge_pos( 100, 240, 400, 240, 50,
                              50, &edge_info );     /*  エッジ位置の計測  */

        /*  計測結果の表示  */
        Lib_sprintf( string, "num:%d", edge_info.num );
        Lib_chrdisp( 1, 1, string );

        for ( i = 0; i < edge_info.num; i++ )
        {
             Lib_sprintf( string, "up x:%7.2lf", (double)edge_info.upx[i] /16.0 );
             Lib_chrdisp( 1, i+2, string );

             Lib_sprintf( string, "down x:%7.2lf", (double)edge_info.downx[i] /16.0 );
             Lib_chrdisp( 25, i+2, string );

        }

        Lib_em_edge_disp();           /* エッジ位置にマークを表示する */

        Lib_display_message( 400, 400, "終了", "エッジ計測" );

        Lib_em_inspection_close();   /* エッジ検査の終了 */
        Lib_freerun();
    }
}
```

Lib_em_inspection_open

機　能

エッジ計測の開始

形　式

```
#include "f_em.h"
int Lib_em_inspection_open( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジ計測を開始します。この関数を実行しなければ、計測は行えません。
必要なワークメモリのサイズはX方向画素数×３２バイトです。

戻り値

処理結果

値	定　　　数	意　　　味
０	NORMAL_RETURN	正常終了しました。
－１	NOT_ENOUGH_MEMORY	ワークメモリが確保できませんでした。

例

エッジ計測プログラム例－１，エッジ計測プログラム例－２を参照してください。

留意事項

ありません。

Lib_em_inspection_close

機　能　エッジ計測の終了

形　式

```
#include "f_em.h"
void Lib_em_inspection_close( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説　エッジ計測を終了します。

戻り値　ありません。

例　エッジ計測プログラム例－１，エッジ計測プログラム例－２を参照してください。

留意事項　ありません。

Lib_em_avr_inspection

機　能

エッジ平均測定

形　式

```
#include "f_em.h"
int Lib_em_avr_inspection( int sx, int sy, int ex, int ey, int diff_p,
                          int level_p, int precise, int xaspect, int yaspect,
                                  EM_AVR_INSPECTION *avr );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジ計測の出力値を物理寸法にしたい場合に使用します。
エッジ計測を行う座標変換係数を求めるため、平均ピッチ及び平均幅を測定します。
測定開始点が黒（低輝度）の場合は黒を背景とみなして、白（高輝度）の部分の幅を測定します。
測定開始点が白の場合は黒の部分の幅を測定します。

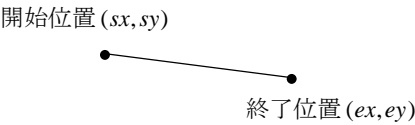
① ***sx*** は測定開始Ｘ位置です。

② ***sy*** は測定開始Ｙ位置です。

③ ***ex*** は測定終了Ｘ位置です。

④ ***ey*** は測定終了Ｙ位置です。

⑤ ***diff_p*** はエッジ検出微分しきい値（１～９９％）です。
仮エッジ検出の際、最大微分値を１００％としてエッジ検出微分しきい値に満たない仮エッジは無視されます。
（「エッジ計測のアルゴリズムについて」の頁で解説しています。）

⑥ ***level_p*** はエッジ検出濃度しきい値（１～９９％）です。
エッジ検出の際、濃度の谷から山までを１００％として、エッジ検出濃度しきい値にあたる座標をエッジ座標としています。
（「エッジ計測のアルゴリズムについて」の頁で解説しています。）

⑦ ***precise*** は精度です。

値	定　　数	意　　味
1	EM_NORMAL_PRECISE	１ラインの測定値を出力
2	EM_HIGH_PRECISE	３ライン測定の平均値を出力
3	EM_SUPER_PRECISE	５ライン測定の平均値を出力

１ラインの測定で結果が安定しない場合は、計測ラインを増すことにより改善されます。

⑧ ***xaspect*** はアスペクト比（横方向）です。

⑨ ***yaspect*** はアスペクト比（縦方向）です。

例えば、

アスペクト比		パラメータ値
１：１	の場合	１０００，１０００とします。
０．９９８１：１	の場合	９９８１，１００００とします。

⑩ ****avr*** は測定結果を格納する EM_AVR_INSPECTION型の構造体へのポインタです。

```
typedef struct
{                           /* エッジ平均測定結果          */
     int     num;           /* 数                          */
     int     pitch;         /* 平均ピッチ（画素の16倍値）  */
     int     width;         /* 平均エッジ幅（画素の16倍値）*/
     int     factor;        /* 座標変換係数                */
} EM_AVR_INSPECTION;
```

戻り値

値	定　　　数	意　　　　味
０	NORMAL_RETURN	正常終了しました。
－１	NOT_ENOUGH_MEMORY	ワークメモリが確保できませんでした。
－２	BAD_POSITION	測定位置が処理範囲を越えています。
－３	NOT_OPEN	オープンしていません。
－４	BAD_PARAMETER	パラメータが異常です。
－５	OUT_OF_SPACE	最大計測数を越えました。

例

エッジ計測プログラム例－１を参照してください。

留意事項

○EM_AVR_INSPECTION型の構造体を確保する必要があります。
　Lib_em_inspection_open を実行前にコールする必要があります。

Lib_em_calib

機　能

座標変換係数を求める

形　式

```
#include "f_em.h"
int Lib_em_calib( EM_AVR_INSPECTION *avr, int pitch, int width );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジ計測の出力値を物理寸法にしたい場合に使用します。
エッジ平均測定で得た測定結果を使用して座標変換係数を求めます。
求められた座標変換係数は構造体に納められます。

① ***avr** は平均測定結果を格納した EM_AVR_INSPECTION型の構造体へのポインタです。

```
typedef struct
{                              /* エッジ平均測定結果              */
     int     num;              /* 数                              */
     int     pitch;            /* 平均ピッチ（画素の16倍値）      */
     int     width;            /* 平均エッジ幅（画素の16倍値）    */
     int     factor;           /* 座標変換係数                    */
} EM_AVR_INSPECTION;
```

② ***pitch*** は変換後のピッチです。
設定範囲は ０～10,000,000 です。

③ ***width*** は変換後のエッジ幅です。
設定範囲は ０～10,000,000 です。

０を設定すると変換係数の算出に反映されません。
pitch, width のどちらかには０以外の値を設定してください。
変換後の値として物理寸法を与える場合は２５６以上の整数を与えます。
例えば、ピッチが１ミリの場合は pitch = 1 とせずに １０００ミクロンとして
pitch = 1000 とします。
これは内部の計算が整数計算なので、数が２５６より少ないと精度が得られなくなるためです。

座標変換係数は、次の方法で算出されます。

(座標変換係数 / 16) = (変換後のピッチ / (平均ピッチ / 16)
+ 変換後の幅 / (平均幅 / 16)) / 2

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－４	BAD_PARAMETER	パラメータが異常です。

例

エッジ計測プログラム例－１を参照してください。

留意事項

○ Lib_em_avr_inspection を実行前にコールする必要があります。

Lib_em_inspection

機　能

エッジ測定

形　式

```
#include "f_em.h"
int Lib_em_inspection( int sx, int sy, int ex, int ey, int diff_p, int level_p,
                       int precise, int xaspect, int yaspect, int factor,
                       EM_INSPECTION *inspection );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

測定結果は、数とピッチと幅及び、中心座標が出力されます。
数は最大５１２本まで測定できます。
測定開始点が黒（低輝度）の場合は黒を背景とみなして、白（高輝度）の部分の幅を測定します。
測定開始点が白の場合は黒の部分の幅を測定します。

① ***sx*** は測定開始Ｘ位置です。

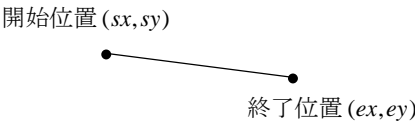
② ***sy*** は測定開始Ｙ位置です。

③ ***ex*** は測定終了Ｘ位置です。

④ ***ey*** は測定終了Ｙ位置です。

開始位置 (*sx*, *sy*)
終了位置 (*ex*, *ey*)

⑤ ***diff_p*** はエッジ検出微分しきい値（１～９９％）です。
仮エッジ検出の際、最大微分値を１００％としてエッジ検出微分しきい値に満たない仮エッジは無視されます。（「エッジ計測のアルゴリズムについて」の頁で解説しています。）

⑥ ***level_p*** はエッジ検出濃度しきい値（１～９９％）です。
エッジ検出の際、濃度の谷から山までを１００％として、エッジ検出濃度しきい値にあたる座標をエッジ座標としています。（「エッジ計測のアルゴリズムについて」の頁で解説しています。）

⑦ ***precise*** は精度です。

値	定　　数	意　　味
1	EM_NORMAL_PRECISE	１ラインの測定値を出力します。
2	EM_HIGH_PRECISE	３ライン測定の平均値を出力します。
3	EM_SUPER_PRECISE	５ライン測定の平均値を出力します。

１ラインの測定で結果が安定しない場合は、測定ラインを増すことにより改善されます。

⑧ ***xaspect*** はアスペクト比（横方向）です。

⑨ ***yaspect*** はアスペクト比（縦方向）です。

例えば、

アスペクト比		パラメータ値
１：１	の場合	１０００，１０００とします。
０．９９８１：１	の場合	９９８１，１００００とします。

⑩ ***factor*** は座標変換係数です。
測定結果を画素単位にしたいときは、factor=256 として出力値を浮動小数点型にして16.0 で割ってください。
測定結果を物理寸法にしたい場合は、あらかじめエッジ平均測定を行い、Lib_em_calib で得た値を factor としてください。

⑪ ****inspection*** は測定結果を格納する EM_INSPECTION型の構造体へのポインタです。

```
#define MAX_NUM    512          /*最大計測可能数*/
typedef struct
{                               /*エッジ測定結果*/
    int     num;                /*数*/
    int     pitch[ MAX_NUM ];   /*ピッチ（座標変換係数で正規化された値）*/
    int     width[ MAX_NUM ];   /*エッジ幅（座標変換係数で正規化された値）*/
    int     x[ MAX_NUM ];       /*中心Ｘ座標（１６倍値） */
    int     y[ MAX_NUM ];       /*中心Ｙ座標（１６倍値） */
} EM_INSPECTION;
```

戻り値

値	定　数	意　味
0	NORMAL_RETURN	正常終了しました。
−１	NOT_ENOUGH_MEMORY	ワークメモリが確保できませんでした。
−２	BAD_POSITION	測定位置が処理範囲を越えています。
−３	NOT_OPEN	オープンしていません。
−４	BAD_PARAMETER	パラメータが異常です。
−５	OUT_OF_SPACE	最大計測数を越えました。

例

エッジ計測プログラム例−１を参照してください。

留意事項

○ EM_INSPECTION型の構造体を確保する必要があります。

Lib_em_edge_pos

機　能

エッジ位置の出力

形　式

```
include "f_em.h"
int Lib_em_edge_pos( int sx, int sy, int ex, int ey,
                     int diff_p, int level_p, EM_EDGE_INFO *edge_info )
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジの位置を測定して出力します。

① ***sx*** は測定開始Ｘ位置です。

② ***sy*** は測定開始Ｙ位置です。

③ ***ex*** は測定終了Ｘ位置です。

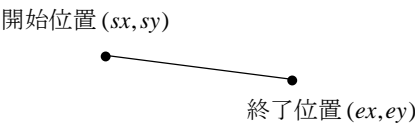
④ ***ey*** は測定終了Ｙ位置です。

⑤ ***diff_p*** はエッジ検出微分しきい値（１～９９％）です。
仮エッジ検出の際、最大微分値を１００％としてエッジ検出微分しきい値に満たない仮エッジは無視されます。（「エッジ計測のアルゴリズムについて」の頁で解説しています。）

⑥ ***level_p*** はエッジ検出濃度しきい値（１～９９％）です。
エッジ検出の際濃度の谷から山までを１００％として、エッジ検出濃度しきい値にあたる座標をエッジ座標としています。（「エッジ計測のアルゴリズムについて」の頁で解説しています。）

⑦ ***edge_info*** はエッジ情報を格納する構造体へのポインタです。

```
#define MAX_NUM          512     /* 最大計測可能数 */
typedef struct
{
    int     num;                 /* 上昇エッジ数及び、下降エッジ数      */
    int     upx[ MAX_NUM ];      /* 濃度値上昇時のＸ位置(16倍値)       */
    int     upy[ MAX_NUM ];      /* 濃度値上昇時のＹ位置(16倍値)       */
    int     downx[ MAX_NUM ];    /* 濃度値下降時のＸ位置(16倍値)       */
    int     downy[ MAX_NUM ];    /* 濃度値下降時のＹ位置(16倍値)       */
} EM_EDGE_INFO;
```

戻り値

値	定　　　数	意　　　味
０	NORMAL_RETURN	正常終了しました。
－５	OUT_OF_SPACE	最大計測数を越えました。

例

エッジ計測プログラム例－２を参照してください。

留意事項

○EM_EDGE_INFO型の構造体を確保する必要があります。
Lib_em_inspection_open を実行前にコールする必要があります。

Lib_em_edge_disp

機　能　　エッジ位置を表示

形　式

```
#include "f_em.h"
void Lib_em_edge_disp( void );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	

解　説　　エッジ位置を線画プレーンに表示します。

戻り値　　ありません。

例　　エッジ計測プログラム例－１，エッジ計測プログラム例－２を参照してください。

留意事項　　○Lib_em_avr_inspection，Lib_em_inspection，Lib_em_edge_pos を実行後、この関数を使用できます。

Lib_em_avr_inspection2

機　能　　エッジ平均測定

形　式

```
#include "f_em.h"
int Lib_em_avr_inspection2( int sx, int sy, int ex, int ey, int diff_p,
                            int level_p, int precise, int xaspect, int yaspect,
                                    int color, EM_AVR_INSPECTION *avr );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジ計測の出力値を物理寸法にしたい場合に使用します。
エッジ計測を行う座標変換係数を求めるため、平均ピッチ及び平均幅を測定します。

Lib_em_avr_inspection との相違点は次の通りです。
　１）計測対象物の色を指定できます。
　２）エッジの検出線の数を１以上の任意の奇数で指定できます。

① ***sx*** は測定開始Ｘ位置です。

② ***sy*** は測定開始Ｙ位置です。

③ ***ex*** は測定終了Ｘ位置です。

④ ***ey*** は測定終了Ｙ位置です。

⑤ ***diff_p*** はエッジ検出微分しきい値（１～９９％）です。
仮エッジ検出の際、最大微分値を１００％としてエッジ検出微分しきい値に満たない仮エッジは無視されます。
（「エッジ計測のアルゴリズムについて」の頁で解説しています。）

⑥ ***level_p*** はエッジ検出濃度しきい値（１～９９％）です。
エッジ検出の際、濃度の谷から山までを１００％として、エッジ検出濃度しきい値にあたる座標をエッジ座標としています。
（「エッジ計測のアルゴリズムについて」の頁で解説しています。）

⑦ ***precise*** はエッジの検出線の数です。１以上の奇数で指定してください。
計測ラインを増すことにより安定性が改善されますが､処理時間は比例して増大します。

⑧ ***xaspect*** はアスペクト比（横方向）です。

⑨ ***yaspect*** はアスペクト比（縦方向）です。

例えば、

アスペクト比		パラメータ値
１：１	の場合	１０００，１０００とします。
０．９９８１：１	の場合	９９８１，１００００とします。

⑩ ***color*** は計測対象物の色です。

値	定　　数	意　　味
0	BLACK_COLOR	計測対象物が黒
1	WHITE_COLOR	計測対象物が白

⑪ ****avr*** は測定結果を格納する EM_AVR_INSPECTION 型の構造体へのポインタです。

```
typedef struct
{                           /* エッジ平均測定結果               */
    int     num;            /* 数                               */
    int     pitch;          /* 平均ピッチ（画素の 16 倍値）     */
    int     width;          /* 平均エッジ幅（画素の 16 倍値）   */
    int     factor;         /* 座標変換係数                     */
} EM_AVR_INSPECTION;
```

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了しました。
－1	NOT_ENOUGH_MEMORY	ワークメモリが確保できませんでした。
－2	BAD_POSITION	測定位置が処理範囲を越えています。
－3	NOT_OPEN	オープンしていません。
－4	BAD_PARAMETER	パラメータが異常です。
－5	OUT_OF_SPACE	最大計測数を越えました。

例

Lib_em_avr_inspection とほぼ同様なので、エッジ計測プログラム例－１を参照してください。

留意事項

○EM_AVR_INSPECTION 型の構造体を確保する必要があります。
　Lib_em_inspection_open を実行前にコールする必要があります。

Lib_em_inspection2

機　能

エッジ測定

形　式

```
#include "f_em.h"
int Lib_em_inspection2( int sx, int sy, int ex, int ey, int diff_p, int level_p,
                        int precise, int xaspect, int yaspect, int factor,
                        int color, EM_INSPECTION *inspection );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

測定結果は、数とピッチと幅及び、中心座標が出力されます。
数は最大５１２本まで測定できます。

Lib_em_inspection との相違点は次の通りです。
　１）計測対象物の色を指定できます。
　２）エッジの検出線の数を１以上の任意の奇数で指定できます。

① ***sx*** は測定開始Ｘ位置です。

② ***sy*** は測定開始Ｙ位置です。

③ ***ex*** は測定終了Ｘ位置です。

④ ***ey*** は測定終了Ｙ位置です。

⑤ ***diff_p*** はエッジ検出微分しきい値（１～９９％）です。
仮エッジ検出の際、最大微分値を１００％としてエッジ検出微分しきい値に満たない仮エッジは無視されます。
（「エッジ計測のアルゴリズムについて」の頁で解説しています。）

⑥ ***level_p*** はエッジ検出濃度しきい値（１～９９％）です。
エッジ検出の際、濃度の谷から山までを１００％として、エッジ検出濃度しきい値にあたる座標をエッジ座標としています。
（「エッジ計測のアルゴリズムについて」の頁で解説しています。）

⑦ ***precise*** はエッジの検出線の数です。１以上の奇数で指定してください。
計測ラインを増すことにより安定性が改善されますが､処理時間は比例して増大します。

⑧ ***xaspect*** はアスペクト比（横方向）です。

⑨ ***yaspect*** はアスペクト比（縦方向）です。

例えば、

アスペクト比		パラメータ値
１：１	の場合	１０００，１０００とします。
０．９９８１：１	の場合	９９８１，１００００とします。

⑩ ***factor*** は座標変換係数です。
測定結果を画素単位にしたいときは、factor=256 として出力値を浮動小数点型にして16.0 で割ってください。
測定結果を物理寸法にしたい場合は、あらかじめエッジ平均測定を行い、Lib_em_calib で得た値を factor としてください。

⑪ ***color*** は計測対象物の色です。

値	定　　数	意　　味
0	BLACK_COLOR	計測対象物が黒
1	WHITE_COLOR	計測対象物が白

⑫ ****inspection*** は測定結果を格納する EM_INSPECTION 型の構造体へのポインタです。

```
#define MAX_NUM    512        /*最大計測可能数*/
typedef struct
{                             /*エッジ測定結果*/
    int    num;               /*数*/
    int    pitch[ MAX_NUM ];  /* ピッチ（座標変換係数で正規化された値）   */
    int    width[ MAX_NUM ];  /* エッジ幅（座標変換係数で正規化された値） */
    int    x[ MAX_NUM ];      /* 中心Ｘ座標（１６倍値）                   */
    int    y[ MAX_NUM ];      /* 中心Ｙ座標（１６倍値）                   */
} EM_INSPECTION;
```

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了しました。
－１	NOT_ENOUGH_MEMORY	ワークメモリが確保できませんでした。
－２	BAD_POSITION	測定位置が処理範囲を越えています。
－３	NOT_OPEN	オープンしていません。
－４	BAD_PARAMETER	パラメータが異常です。
－５	OUT_OF_SPACE	最大計測数を越えました。

例

Lib_em_inspection とほぼ同様なので、エッジ計測プログラム例－１を参照してください。

留意事項

○ EM_INSPECTION 型の構造体を確保する必要があります。

Lib_em_edge_pos2

機　能

エッジ位置の出力

形　式

```
include "f_em.h"
int Lib_em_edge_pos2( int sx, int sy, int ex, int ey, int diff_p, int level_p,
                      int precise, int color, EM_EDGE_INFO *edge_info );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

エッジの位置を測定して出力します。

Lib_em_edge_pos との相違点は次の通りです。
　１）計測対象物の色を指定できます。
　２）エッジの検出線の数を１以上の任意の奇数で指定できます。

① ***sx*** は測定開始Ｘ位置です。

② ***sy*** は測定開始Ｙ位置です。

③ ***ex*** は測定終了Ｘ位置です。

④ ***ey*** は測定終了Ｙ位置です。

⑤ ***diff_p*** はエッジ検出微分しきい値（１～９９％）です。
仮エッジ検出の際、最大微分値を１００％としてエッジ検出微分しきい値に満たない仮エッジは無視されます。
（「エッジ計測のアルゴリズムについて」の頁で解説しています。）

⑥ ***level_p*** はエッジ検出濃度しきい値（１～９９％）です。
エッジ検出の際濃度の谷から山までを１００％として、エッジ検出濃度しきい値にあたる座標をエッジ座標としています。
（「エッジ計測のアルゴリズムについて」の頁で解説しています。）

⑦ ***precise*** はエッジの検出線の数です。１以上の奇数で指定してください。
計測ラインを増すことにより安定性が改善されますが､処理時間は比例して増大します。

⑧ ***color*** は計測対象物の色です。

値	定　　数	意　　味
0	BLACK_COLOR	計測対象物が黒
1	WHITE_COLOR	計測対象物が白

⑨ ***edge_info*** はエッジ情報を格納する構造体へのポインタです。

```
#define MAX_NUM           512     /* 最大計測可能数 */
typedef struct
{
    int     num;                  /* 上昇エッジ数及び、下降エッジ数     */
    int     upx[ MAX_NUM ];       /* 濃度値上昇時のＸ位置(16 倍値)      */
    int     upy[ MAX_NUM ];       /* 濃度値上昇時のＹ位置(16 倍値)      */
    int     downx[ MAX_NUM ];     /* 濃度値下降時のＸ位置(16 倍値)      */
    int     downy[ MAX_NUM ];     /* 濃度値下降時のＹ位置(16 倍値)      */
} EM_EDGE_INFO;
```

戻り値

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	NOT_ENOUGH_MEMORY	ワークメモリが確保できませんでした。
－２	BAD_POSITION	測定位置が処理範囲を越えています。
－３	NOT_OPEN	オープンしていません。
－４	BAD_PARAMETER	パラメータが異常です。
－５	OUT_OF_SPACE	最大計測数を越えました。

例

Lib_em_edge_pos とほぼ同様なので、エッジ計測プログラム例－２を参照してください。

留意事項

○EM_EDGE_INFO 型の構造体を確保する必要があります。
　Lib_em_inspection_open を実行前にコールする必要があります。

エッジ計測のアルゴリズムについて

１．濃淡エッジ計測ライブラリにおけるエッジ検出の前提条件

濃淡画像ライブラリの「濃淡エッジ計測ライブラリ」は、以下の図１，図２のように、計測ライン上に濃度の山（始点が暗なら終点も暗）、または濃度の谷（始点が明なら終点も明）が１組以上存在するというような条件(例えば IC の足の部分)でのエッジ計測に適しています。

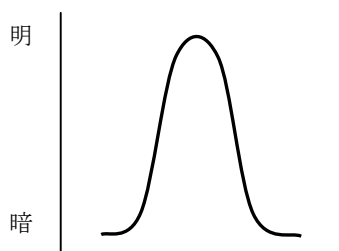
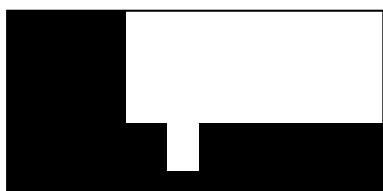
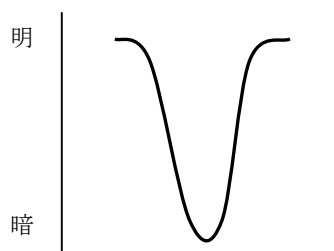
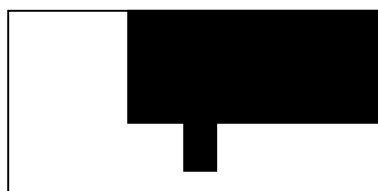



図１　　図２

図３，図４のような明るくなりっぱなしや、暗くなりっぱなしの状態では、エッジが検出できない、または誤検出する場合があります。

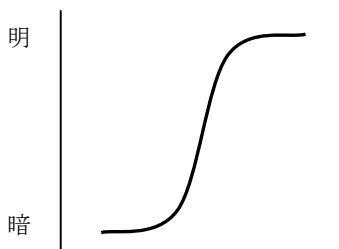
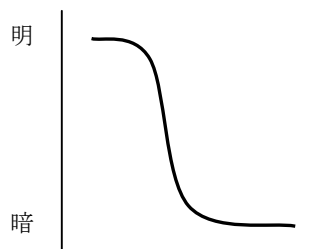
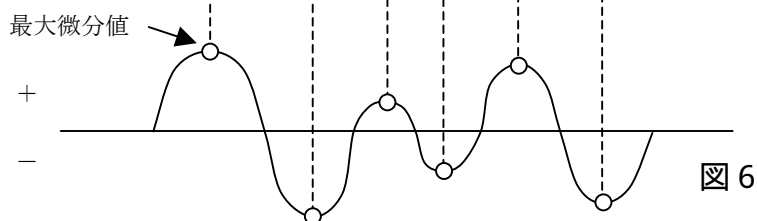
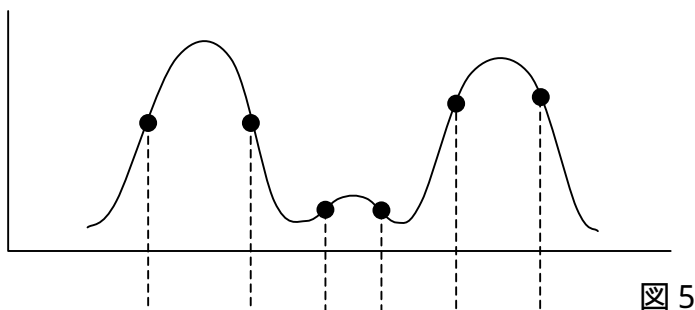



図３　　図４

２．エッジ計測のアルゴリズムについて

検査ライン上の濃度波形が図５のようだとします。まず、検査ライン上の濃度値を微分します。
微分(厳密に言えば差分ですが)とは１画素おきに濃度値を参照し引き算することです。
計測ラインの始点座標が(50, 100)、終点座標が(100, 100)だとすると、(52, 100)の濃度値から(50, 100)の濃度値を引く、(53, 100)の濃度値から(51, 100)の濃度値を引く・・・、という処理を繰り返します。求めた微分値をプロットすると図６のような波形になります。
微分値から微分波形の頂点(図６の白丸)がわかります。そして、最大微分値もわかります。

図５
図６

次に仮のエッジ位置を検出します。

Δd　：微分値
$\Delta d \max$　：最大微分値
α　：微分しきい値（%）

とすると

$$\Delta d \geqq \Delta d \max \times \alpha / 100$$

となった点が仮のエッジ点(図７の黒丸)となります。

図7

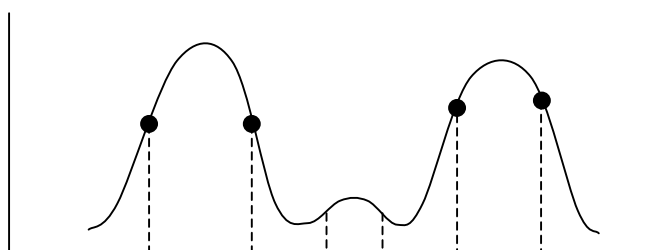
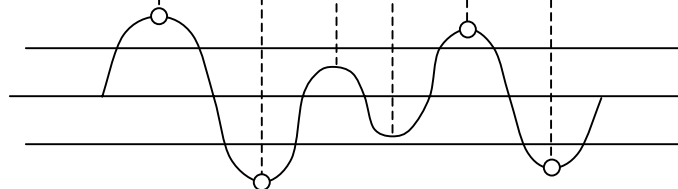



図8

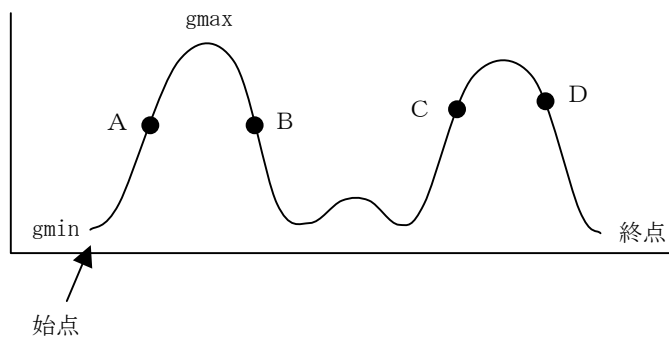



図9

次に、実際のエッジ点の座標を求めます。仮のエッジ位置Ａのエッジ座標を求めるために、始点から仮エッジＢまでの間の最大濃度値、最小濃度値を求めます。図９のように始点位置の濃度値が $g\min$ でＡとＢの間に $g\max$ があります。

ここで、

$g\max$ ：最大濃度値

$g\min$ ：最小濃度値

β ：濃度しきい値（%）

とすると、

$$(g\max - g\min) \times \beta / 100$$

の濃度をもつエッジ座標が求められます。

同様な処理で仮のエッジ位置Ｂのエッジ座標を求めます。

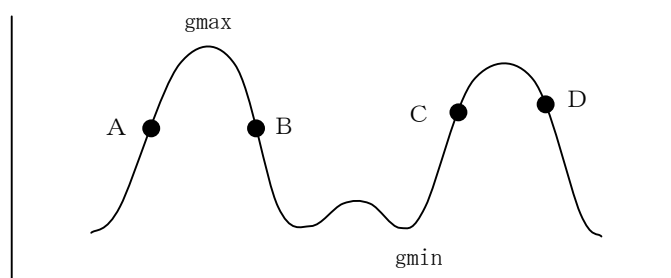



図１０

ＡからＣの間の最大濃度値、最小濃度値を求め、同様な計算を行うことでＢのエッジの座標が求められます。

8．画像強調・フィルタリングライブラリ

本ライブラリは、濃淡画像メモリの画像強調あるいはフィルタリングを行うものです。
本ライブラリを使用する場合、濃淡画像メモリが最低２面必要になります。

Lib_averaging

機　能

近傍平均

形　式

```
#include "f_filter.h"
void Lib_averaging ( int  src_mem_no, int dst_mem_no );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

指定ウィンドウ内の８ビットの深さを持つ濃淡画像において、各点の濃度値をその点の周囲の点の濃度値の平均値とすることにより、雑音除去を行います。
この平均値を求める時の近傍点数は８点です。
注目する点の出力濃度値を a_0 ，近傍点の濃度値を a_n $(n=1\sim8)$ とすると、下記の様になります。

a_4	a_3	a_2
a_5	a_0	a_1
a_6	a_7	a_8

$$a_0 = \frac{\sum_{i=1}^{8} a_i}{8}$$

① ***src_mem_no*** は、処理対象メモリ番号です。

② ***dst_mem_no*** は、結果格納メモリ番号です。
　src_mem_no とは、異なるメモリ番号を指定してください。

戻り値

ありません。

例

平均化後、モニタＴＶ上に表示します。

```
#include "f_filter.h"
#include "f_video.h"

#define  SRC_MEM_NO     0
#define  DST_MEM_NO     1

void ave_disp()
{
    Lib_freeze( TRANSMIT );
    Lib_averaging( SRC_MEM_NO, DST_MEM_NO );
    Lib_xvideo_transmit( DST_MEM_NO, GRAY_PLANE );
}
```

留意事項

○処理範囲

ステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、使用します。

○外周画素の未処理について

四周１画素ずつ、それぞれ指定ウィンドウの外周データは処理されません。

Lib_laplacian

機　能

ラプラシアン

形　式

```
#include "f_filter.h"
void Lib_laplacian ( int src_mem_no, int dst_mem_no )
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

指定ウィンドウ内の８ビット深さを持つ濃淡画像について、各点の濃度値をその近傍点との差分値と置き換え、画像を尖鋭化するものです。
この差分値をもとめる時の近傍点数は８点です。

注目する点の濃度値を a_0 、近傍点の濃度値を a_1 ～ a_8 とすると、 a_0 の出力濃度は下記のものとなります。

1	1	1
1	－8	1
1	1	1

a_0 の出力濃度

$$= a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8 - 8 \cdot a_0$$

① ***src_mem_no*** は、処理対象メモリ番号です。

② ***dst_mem_no*** は、結果格納メモリ番号です。
　src_mem_no とは、異なるメモリ番号を指定してください。

戻り値

ありません。

例

ラプラシアン後、モニタＴＶ上に表示します。

```
#include "f_filter.h"
#include "f_video.h"

#define  SRC_MEM_NO     0
#define  DST_MEM_NO     1

void lap_disp()
{
    Lib_freeze( TRANSMIT );
    Lib_laplacian( SRC_MEM_NO, DST_MEM_NO );
    Lib_xvideo_transmit( DST_MEM_NO, GRAY_PLANE );
}
```

留意事項

○処理範囲
　ステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、使用します。

○計算結果の精度について
　差分値の計算結果が２５５を越える場合は、２５５が画像メモリにセットされます。

○外周画素の未処理について
　四周１画素ずつ、それぞれ指定ウィンドウの外周データは処理されません。

Lib_max_filter

機　能

近傍最大値

形　式

```
#include "f_filter.h"
void Lib_max_filter (int src_mem_no, int  dst_mem_no );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

指定ウィンドウ内の８ビット深さを持つ濃淡画像について、各点の濃度値をその近傍点の最大値と置き換え、画像を拡大化するものです。
この最大値をもとめる時の近傍点数は８点です。
注目する点の濃度値を a_0 近傍点の濃度値を a_1 ～ a_8 とすると、a_0 の出力濃度は下記のものとなります。

a_4	a_3	a_2
a_5	a_0	a_1
a_6	a_7	a_8

$$a_0 = \max_{全 i} (a_i)$$

（ a_0 の出力濃度は、 a_0 ～ a_8 の濃度の小さい順に並べ、最大値を a_0 の濃度にする。）

① ***src_mem_no*** は、処理対象メモリ番号です。

② ***dst_mem_no*** は、結果格納メモリ番号です。
　src_mem_noとは、異なるメモリ番号を指定してください。

戻り値

ありません。

例

近傍最大値後、モニタＴＶ上に表示します。

```
#include "f_filter.h"
#include "f_video.h"

#define  SRC_MEM_NO    0
#define  DST_MEM_NO    1

void max_disp()
{
    Lib_freeze( TRANSMIT );
    Lib_max_filter( SRC_MEM_NO, DST_MEM_NO );
    Lib_xvideo_transmit( DST_MEM_NO, GRAY_PLANE );
}
```

留意事項

○処理範囲

ステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、使用します。

○外周画素の未処理について

四周１画素ずつ、それぞれ指定ウィンドウの外周データは処理されません。

Lib_min_filter

機　能

近傍最小値

形　式

```
#include "f_filter.h"
void Lib_min_filter ( int src_mem_no, int dst_mem_no );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

指定ウィンドウ内の８ビット深さを持つ濃淡画像について、各点の濃度値をその近傍点の最小値と置き換え、画像を縮小化するものです。
この最小値をもとめる時の近傍点数は８点です。
注目する点の濃度値を a_0 近傍点の濃度値を a_1 ～ a_8 とすると、a_0 の出力濃度は下記のものとなります。

a_4	a_3	a_2
a_5	a_0	a_1
a_6	a_7	a_8

$$a_0 = \min_{全 i} (a_i)$$

（ a_0 の出力濃度は、 a_0 ～ a_8 の濃度の小さい順に並べ、最大値を a_0 の濃度にする。）

① ***src_mem_no*** は、処理対象メモリ番号です。

② ***dst_mem_no*** は、結果格納メモリ番号です。
src_mem_no とは、異なるメモリ番号を指定してください。

戻り値

ありません。

例

近傍最小値後、モニタＴＶ上に表示します。

```
#include "f_filter.h"
#include "f_video.h"

#define  SRC_MEM_NO     0
#define  DST_MEM_NO     1

void min_disp()
{
    Lib_freeze( TRANSMIT );
    Lib_min_filter( SRC_MEM_NO, DST_MEM_NO );
    Lib_xvideo_transmit( DST_MEM_NO, GRAY_PLANE );
}
```

留意事項

○処理範囲

ステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、使用します。

○外周画素の未処理について

四周１画素ずつ、それぞれ指定ウィンドウの外周データは処理されません。

Lib_max4_filter

機　能

４近傍最大値

形　式

```
#include "f_filter.h"
int Lib_max4_filter( int src_mem_no, int dst_mem_no, int mode );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

指定ウィンドウ内の８ビットの深さを持つ濃淡画像について、各点の濃度値をその近傍点の最大値と置き換え、画像を拡大化するものです。
この最大値を求めるときの近傍点数は４点です。
a_0 近傍点において注目する点 a_0 ～ a_4 の濃度値より最大値を求め、a_0 の濃度値をその最大値と置き換えます。

	a_1	
a_2	a_0	a_4
	a_3	

① ***src_mem_no*** は、処理対象メモリ番号です。

② ***dst_mem_no*** は、結果格納メモリ番号です。
　src_mem_no とは、異なるメモリ番号を指定してください。

③ ***mode*** は、処理範囲を指定する番号です。

値	意　　味
0	点 a_0 ～ a_4 の濃度値より最大値を求めます。
1	点 a_0, a_2, a_4 の濃度値より最大値を求めます。
2	点 a_0, a_1, a_3 の濃度値より最大値を求めます。

戻り値

処理結果

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了しました。
－1	ERROR_RETURN	異常終了しました。

例

処理後、結果をモニタＴＶに表示します。

```
#include "f_filter.h"
#include "f_video.h"

#define  SRC_MEM_NO    0
#define  DST_MEM_NO    1
#define  MODE          0

void max4_disp()
{
        Lib_freeze( TRANSMIT );
        Lib_max4_filter( SRC_MEM_NO, DST_MEM_NO, MODE );
        Lib_xvideo_transmit( DST_MEM_NO, GRAY_PLANE );
}
```

留意事項

○処理範囲

ステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、使用します。

○外周画素の未処理について

① mode　0
　四周１画素ずつ、それぞれ指定ウィンドウの外周データは処理されません。

② mode　1
　左右両端１画素ずつ、それぞれ指定ウィンドウの外周データは処理されません。

③ mode　2
　上下両端１画素ずつ、それぞれ指定ウィンドウの外周データは処理されません。

Lib_min4_filter

機　能

４近傍最小値

形　式

```
#include "f_filter.h"
int Lib_min4_filter( int src_mem_no, int dst_mem_no, int mode );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

指定ウィンドウ内の８ビットの深さを持つ濃淡画像について、各点の濃度値をその近傍点の最小値と置き換え、画像を縮小化するものです。
この最小値を求めるときの近傍点数は４点です。
a_0 近傍点において注目する a_0 ～ a_4 の濃度値より最小値を求め、a_0 の濃度値をその最小値と置き換えます。

	a_1	
a_2	a_0	a_4
	a_3	

① ***src_mem_no*** は、処理対象メモリ番号です。

② ***dst_mem_no*** は、結果格納メモリ番号です。
src_mem_no とは、異なるメモリ番号を指定してください。

③ ***mode*** は、処理範囲を指定する番号です。

値	意　　味
0	点 a_0 ～ a_4 の濃度値より最小値を求めます。
1	点 a_0 , a_2 , a_4 の濃度値より最小値を求めます。
2	点 a_0 , a_1 , a_3 の濃度値より最小値を求めます。

戻り値

処理結果

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了しました。
－1	ERROR_RETURN	異常終了しました。

例

処理後、結果をモニタＴＶに表示します。

```
#include "f_filter.h"
#include "f_video.h"

#define  SRC_MEM_NO    0
#define  DST_MEM_NO    1
#define  MODE          0

void min4_disp()
{
        Lib_freeze( TRANSMIT );
        Lib_min4_filter( SRC_MEM_NO, DST_MEM_NO, MODE );
        Lib_xvideo_transmit( DST_MEM_NO, GRAY_PLANE );
}
```

留意事項

○処理範囲

ステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、使用します。

○外周画素の未処理について

① mode ０
四周１画素ずつ、それぞれ指定ウィンドウの外周データは処理されません。

② mode １
左右両端１画素ずつ、それぞれ指定ウィンドウの外周データは処理されません。

③ mode ２
上下両端１画素ずつ、それぞれ指定ウィンドウの外周データは処理されません。

Lib_median

機　能

メディアン

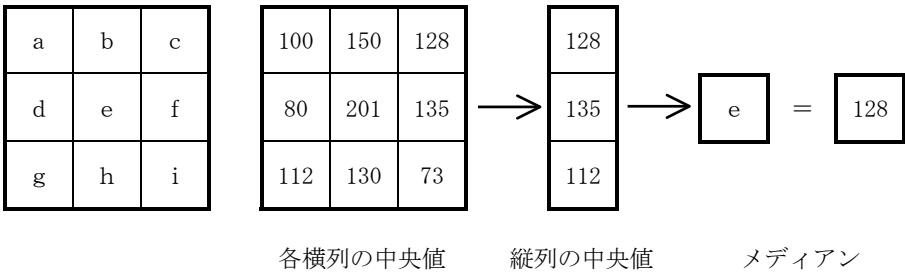
形　式

```
#include "f_filter.h"
int Lib_median ( int src_mem_no, int dst_mem_no );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

８近傍のメディアンフィルタを行います。
近傍画素の値を大小順に並べて真中の値（メディアン）を求めるものです。
しかし、完全なメディアンフィルタは処理時間が秒単位になってしまうので、近似メディアンフィルタを使用しています。
メディアンフィルタは、エッジなどの画像の重要な情報を損なうことなく、特にスパイク状の雑音を取り除くことができます。
またなめらかなエッジに対しては、その形状をそのまま保存する特徴があります。

（例）近似メディアンフィルタのアルゴリズム

① ***src_mem_no*** は処理対象メモリＮｏ．です。

② ***dst_mem_no*** は結果格納メモリＮｏ．です。
　src_mem_no とは、異なるメモリ番号を指定してください。

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

メディアンフィルタを実施し、結果を表示します。

```
#include  "f_gui.h"
#include  "f_video.h"
#include  "f_pinf.h"
#include  "f_stdlib.h"
#include  "f_filter.h"


int disp()
{
    int dst_mem_no;
    int status;

    status = ERROR_RETURN;
    Lib_init_cursor();
    if( ERROR_RETURN != (dst_mem_no = Lib_alloc_gray_memory()) )
    {
        Lib_freeze( TRANSMIT );
        status = Lib_median( Lib_get_gray_memory(), dst_mem_no );
        if( NORMAL_RETURN == status )
        Lib_xvideo_transmit( dst_mem_no, GRAY_PLANE );
        Lib_display_keyinput( 10,450, "入力待ﾁ");
        Lib_freerun();
    }
    return( status );
}
```

留意事項

○処理範囲
ステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、使用します。

○外周１画素は出力されません。

Lib_roberts

機　能

微分　Ｒｏｂｅｒｔｓ　オペレータ

形　式

```
#include "f_filter.h"
int Lib_roberts ( int src_mem_no, int dst_mem_no );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

微分の Roberts オペレータは２×２のエッジ抽出オペレータです。
斜め方向の勾配を与えるもので、斜めのエッジ検出に有効です。
２×２局所並列演算によって、次式に示される画像データの空間１次微分（グラジエント）を計算します。

$$g(i,j)=\left|f(i,j)-f(i+1,j+1)\right|+\left|f(i+1,j)-f(i,j+1)\right|$$

ここで、 f は原画像、 g は処理画像を示します。

① ***src_mem_no*** は処理対象メモリＮｏ．です。

② ***dst_mem_no*** は結果格納メモリＮｏ．です。
src_mem_no とは、異なるメモリ番号を指定してください。

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

微分 roberts オペレータを実施し、結果を表示します。

```
#include  "f_gui.h"
#include  "f_video.h"
#include  "f_pinf.h"
#include  "f_stdlib.h"
#include  "f_filter.h"


int disp()
{
    int dst_mem_no;
    int status;

    status = ERROR_RETURN;
    Lib_init_cursor();
    if( ERROR_RETURN != (dst_mem_no = Lib_alloc_gray_memory()) )
    {
        Lib_freeze( TRANSMIT );
        status = Lib_roberts( Lib_get_gray_memory(), dst_mem_no );
        if( NORMAL_RETURN == status )
        Lib_xvideo_transmit( dst_mem_no, GRAY_PLANE );
        Lib_display_keyinput( 10,450, "入力待ﾁ");
        Lib_freerun();
    }
     return( status );
}
```

留意事項

○処理範囲
ステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、使用します。

○ＸＹ終端１画素は出力されません。

○結果は絶対値としています。

○濃度値が２５５を越えた場合は２５５とします。

Lib_sobel

機　能　　微分　Ｓｏｂｅｌ　オペレータ

形　式

```
#include "f_filter.h"
int Lib_sobel ( int src_mem_no, int dst_mem_no, int direction );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

微分の Sobel オペレータは３×３（８近傍）のエッジ抽出オペレータです。
Robertsオペレータと比較して、処理時間は遅いですが、よりエッジを強調する事ができます。
中央画素の周囲の画素の濃度値に係数を掛け、加算したものを中央画素の濃度値とします。

簡略表現

Ｘ方向

$$xg(i,j) = f(i-1,j-1) + 2f(i-1,j) + f(i-1,j+1) - \{ f(i+1,j-1) + 2f(i+1,j) + f(i+1,j+1) \}$$

1	0	−1
2	0	−2
1	0	−1

Ｙ方向

$$yg(i,j) = f(i-1,j-1) + 2f(i,j-1) + f(i+1,j+1) - \{ f(i-1,j+1) + 2f(i,j+1) + f(i+1,j+1) \}$$

1	2	1
0	0	0
−1	−2	−1

ＸＹ方向

$$g(i,j) = |xg(i,j)| + |yg(i,j)|$$

ここで、 f は原画像、 xg , yg , g は処理画像を示します。

① ***src_mem_no*** は処理対象メモリＮｏ．です。

② ***dst_mem_no*** は結果格納メモリＮｏ．です。
src_mem_no とは、異なるメモリ番号を指定してください。

③ ***direction*** は検出方向です。

値	定　数	意　味
0	X_DIRECTION	Ｘ方向を計算します。
1	Y_DIRECTION	Ｙ方向を計算します。
2	XY_DIRECTION	ＸＹ方向を計算します。

戻り値

処理結果

値	定　数	意　味
0	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

微分 sobel オペレータ（ＸＹ方向）を実施し、結果を表示します。

```
#include  "f_gui.h"
#include  "f_video.h"
#include  "f_pinf.h"
#include  "f_stdlib.h"
#include  "f_filter.h"


int disp()
{
    int dst_mem_no;
    int status;

    status = ERROR_RETURN;
    Lib_init_cursor();
    if( ERROR_RETURN != (dst_mem_no = Lib_alloc_gray_memory()) )
    {
        Lib_freeze( TRANSMIT );
        status = Lib_sobel( Lib_get_gray_memory(), dst_mem_no, XY_DIRECTION );
        if( NORMAL_RETURN == status )
        Lib_xvideo_transmit( dst_mem_no, GRAY_PLANE );
        Lib_display_keyinput( 10,450, "入力待ﾁ");
        Lib_freerun();
    }
    return( status );
}
```

留意事項

○処理範囲
ステージパラメータ(システムパラメータ)の「処理ウィンドウ」を参照し、使用します。

○外周１画素は出力されません。

○結果は絶対値としています。

○濃度値が２５５を越えた場合は２５５とします。

Lib_fdefferential

機　能

１次微分　オペレータ

形　式

```
#include "f_filter.h"
int Lib_fdefferential ( int src_mem_no, int dst_mem_no, int direction );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

１次微分オペレータは３×３（８近傍）のエッジ抽出オペレータです。
中央画素の周囲の画素の濃度値に係数を掛け、加算したものを中央画素の濃度値とします。

簡略表現

Ｘ方向

$$xg(i,j)=f(i-1,j-1)+f(i-1,j)+f(i-1,j+1)$$
$$-\{f(i+1,j-1)+f(i+1,j)+f(i+1,j+1)\}$$

1	0	−1
1	0	−1
1	0	−1

Ｙ方向

$$yg(i,j)=f(i-1,j-1)+f(i,j-1)+f(i+1,j+1)$$
$$-\{f(i-1,j+1)+f(i,j+1)+f(i+1,j+1)\}$$

1	1	1
0	0	0
−1	−1	−1

ＸＹ方向

$$g(i,j)=|xg(i,j)|+|yg(i,j)|$$

ここで、 f は原画像、 xg , yg , g は処理画像を示します。

① ***src_mem_no*** は処理対象メモリＮｏ．です。

② ***dst_mem_no*** は結果格納メモリＮｏ．です。
src_mem_no とは、異なるメモリ番号を指定してください。

③ ***direction*** は検出方向です。

値	定　　数	意　　味
0	X_DIRECTION	Ｘ方向を計算します。
1	Y_DIRECTION	Ｙ方向を計算します。
2	XY_DIRECTION	ＸＹ方向を計算します。

戻り値

処理結果

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

１次微分 オペレータ（ＸＹ方向）を実施し、結果を表示します。

```
#include  "f_gui.h"
#include  "f_video.h"
#include  "f_pinf.h"
#include  "f_stdlib.h"
#include  "f_filter.h"


int disp()
{
    int dst_mem_no;
    int status;

    status = ERROR_RETURN;
    Lib_init_cursor();
    if( ERROR_RETURN != (dst_mem_no = Lib_alloc_gray_memory()) )
    {
        Lib_freeze( TRANSMIT );
        status = Lib_fdefferential( Lib_get_gray_memory(),
                                              dst_mem_no, XY_DIRECTION );
        if( NORMAL_RETURN == status )
        Lib_xvideo_transmit( dst_mem_no, GRAY_PLANE );
        Lib_display_keyinput( 10,450, "入力待ﾁ");
        Lib_freerun();
    }
    return( status );
}
```

留意事項

○処理範囲
　ステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、使用します。

○外周１画素は出力されません。

○結果は絶対値としています。

○濃度値が２５５を越えた場合は２５５とします。

Lib_sdefferential

機　能

２次微分　オペレータ

形　式

```
#include "f_filter.h"
int Lib_sdefferential ( int src_mem_no, int dst_mem_no );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

２次微分オペレータは３×３（８近傍）のエッジ抽出オペレータです。
中央画素の周囲の画素の濃度値に係数を掛け、加算したものを中央画素の濃度値とします。

簡略表現

$$g(i,j)=f(i-1,j-1)+f(i-1,j)+f(i-1,j+1)+f(i,j-1)+f(i,j+1)+f(i+1,j-1)+f(i+1,j)+f(i+1,j+1)-8f(i,j)$$

1	1	1
1	−8	1
1	1	1

ここで、 f は原画像、 g は処理画像を示します。

① ***src_mem_no*** は処理対象メモリＮｏ．です。

② ***dst_mem_no*** は結果格納メモリＮｏ．です。
　src_mem_no とは、異なるメモリ番号を指定してください。

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

２次微分　オペレータを実施し、結果を表示します。

```
#include  "f_gui.h"
#include  "f_video.h"
#include  "f_pinf.h"
#include  "f_stdlib.h"
#include  "f_filter.h"

int disp()
{
    int dst_mem_no;
    int status;

    status = ERROR_RETURN;
    Lib_init_cursor();
    if( ERROR_RETURN != (dst_mem_no = Lib_alloc_gray_memory()) )
    {
        Lib_freeze( TRANSMIT );
        status = Lib_sdefferential( Lib_get_gray_memory(), dst_mem_no );
        if( NORMAL_RETURN == status )
        Lib_xvideo_transmit( dst_mem_no, GRAY_PLANE );
        Lib_display_keyinput( 10,450, "入力待ﾁ");
        Lib_freerun();
    }
    return( status );
}
```

留意事項

○処理範囲
　ステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、使用します。

○外周１画素は出力されません。

○結果は絶対値としています。

○濃度値が２５５を越えた場合は２５５とします。

○ Lib_laplacian との相違は、Lib_sdefferential が絶対値を使用しているのに対し、前者はマイナスの場合構わず、０を代入している点です。

Lib_sharp

機　能

鮮鋭化

形　式

```
#include "f_filter.h"
int Lib_sharp ( int src_mem_no, int dst_mem_no );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

３×３（８近傍）の鮮鋭化を行います。
画質の悪い画像に対して行われるものですが、雑音成分も同時に強調してしまうという欠点があります。
中央画素の周囲の画素の濃度値に係数を掛け、加算したものを中央画素の濃度値とします。

簡略表現

$$
\begin{aligned}
g(i,j) = 9f(i,j) - \{ & f(i-1,j-1) + f(i,j-1) \\
& + f(i+1,j-1) + f(i-1,j) + f(i+1,j) \\
& f(i-1,j+1) + f(i,j+1) + f(i+1,j+1) \}
\end{aligned}
$$

−1	−1	−1
−1	9	−1
−1	−1	−1

ここで、 f は原画像、 g は処理画像を示します。

① ***src_mem_no*** は処理対象メモリＮｏ．です。

② ***dst_mem_no*** は結果格納メモリＮｏ．です。
src_mem_no とは、異なるメモリ番号を指定してください。

戻り値

処理結果

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

鮮鋭化を実施し、結果を表示します。

```
#include  "f_gui.h"
#include  "f_video.h"
#include  "f_pinf.h"
#include  "f_stdlib.h"
#include  "f_filter.h"


int disp()
{
    int dst_mem_no;
    int status;

    status = ERROR_RETURN;
    Lib_init_cursor();
    if( ERROR_RETURN != (dst_mem_no = Lib_alloc_gray_memory()) )
    {
        Lib_freeze( TRANSMIT );
        status = Lib_sharp( Lib_get_gray_memory(), dst_mem_no );
        if( NORMAL_RETURN == status )
        Lib_xvideo_transmit( dst_mem_no, GRAY_PLANE );
        Lib_display_keyinput( 10,450, "入力待ﾁ");
        Lib_freerun();
    }
    return( status );
}
```

留意事項

○処理範囲
　ステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、使用します。

○外周１画素は出力されません。

○結果は絶対値としています。

○濃度値が２５５を越えた場合は２５５とします。

Lib_get_convolver

機　能

ラプラシアン ガウシアンオペレータの係数取得

形　式

```
#include  "f_filter.h"
int Lib_get_convolver( int size, double sigma, int *op, int *mult );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

下に示す計算式によってラプラシアン ガウシアンのコンボルバーを作成します。

$$\nabla^2 G(x,y) = \frac{x^2 + y^2 - 2\sigma^2}{2\pi\sigma^6} \exp\left(- \frac{x^2 + y^2}{2\sigma^2} \right)$$

① ***size*** は、コンボルバーのサイズです。３以上の奇数値を設定します。

② ***sigma*** は、シグマ値です。０より大きい実数を指定します。

③ ****op*** は、コンボルバーへのポインターです。
呼び出し側で要素数が$size^2$の配列を定義しておいてください。
（例：①の size が３なら９の配列）

④ ****mult*** は、係数の倍率値です。
一般的に係数は実数になりますので整数として扱うため、この倍率を掛けています。

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

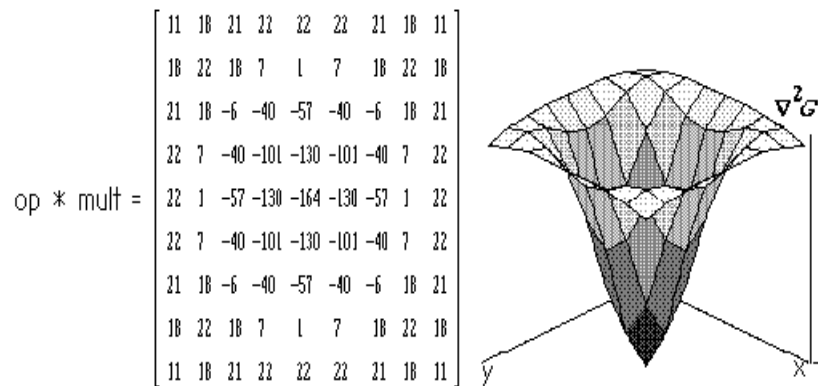
例

サイズ＝９，σ＝２．１のオペレータの係数を取得します。

```
#include "f_filter.h"
#define  SIZE  9
#define  SIGMA 2.1
int test();
{
    int *op;
    int mult;
    int status;
    if ( NULL != ( op =(int *)Lib_mlaloc( sizeof(int) * SIZE * SIZE)) )
    {
        status = Lib_get_convolver( SIZE, SIGMA, op, &mult);
        /*   処理の実行  */
        Lib_lfree( (char *)op );
    }
    return status;
}
```

mult = 10000, コンボルバーは、下のようになります。

$$
op * mult = \begin{bmatrix}
11 & 18 & 21 & 22 & 22 & 22 & 21 & 18 & 11 \\
18 & 22 & 18 & 7 & 1 & 7 & 18 & 22 & 18 \\
21 & 18 & -6 & -40 & -57 & -40 & -6 & 18 & 21 \\
22 & 7 & -40 & -101 & -130 & -101 & -40 & 7 & 22 \\
22 & 1 & -57 & -130 & -164 & -130 & -57 & 1 & 22 \\
22 & 7 & -40 & -101 & -130 & -101 & -40 & 7 & 22 \\
21 & 18 & -6 & -40 & -57 & -40 & -6 & 18 & 21 \\
18 & 22 & 18 & 7 & 1 & 7 & 18 & 22 & 18 \\
11 & 18 & 21 & 22 & 22 & 22 & 21 & 18 & 11
\end{bmatrix}
$$

$\nabla^2 G$

留意事項

○ここで得られた係数値を Lib_lg_filter の引数（*op, mult)として使用します。

Lib_lg_filter

機　能

ラプラシアン ガウシアンオペレータ

形　式

```
#include "f_filter.h"
int  Lib_lg_filter(int src_mem_no, int dst_mem_no,
                                    int *op, int mult, int size);
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

指定ウインドウ内の８ビットの深さを持つ濃淡画像において、ガウス関数と２次関数とが同時に行えるコンボルバーで畳み込み演算をおこない、エッジ抽出をするものです。

① ***src_mem_no*** は、処理対象メモリ番号です。

② ***dst_mem_no*** は、結果格納メモリ番号です。
　src_mem_no とは、異なるメモリ番号を指定してください。

③ ****op*** は、コンボルバーへのポインターです。
　その内容は Lib_get_convolver で得られたものになります。

④ ***mult*** は、コンボルバーの倍率値です。Lib_get_convolver で得られた数値です。

⑤ ***size*** は、コンボルバーの大きさです。
　Lib_get_convolver で使った数値を入れてください。

戻り値

処理結果

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

処理後、モニタＴＶ上に表示します。

```
#include  "f_gui.h"
#include  "f_video.h"
#include  "f_pinf.h"
#include  "f_stdlib.h"
#include  "f_filter.h"

#define SIGMA 3.5
#define SIZE  11

int disp()
{
    int dst_mem_no;
    int status;
    int *op;
    int mult;

    Lib_init_cursor();
    if( NULL != ( op = ( int *)Lib_mlalloc( sizeof( int ) * SIZE * SIZE )))
    {
        if ( NORMAL_RETURN == ( status = Lib_get_convolver
                                     ( SIZE, SIGMA, op, &mult)) )
        {
             if ( ERROR_RETURN != (dst_mem_no = Lib_alloc_gray_memory()) )
             {
                  Lib_freeze( TRANSMIT );
                  status = Lib_lg_filter
                         ( Lib_get_gray_memory(), dst_mem_no, op, mult, SIZE);
                  if ( NORMAL_RETURN == status )
                       Lib_xvideo_transmit( dst_mem_no, GRAY_PLANE );
                  Lib_display_keyinput( 10, 450, "入力待ち");
                  Lib_freerun();
             }
        }
        Lib_lfree( ( char *)op);
    }
    return ( status );
}
```

留意事項

○処理範囲はステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、
使用します。
また、それぞれ指定ウィンドウの外周の(N/2-1)画素は処理されません。

○ size が３および５の時は、特殊な処理を行う事によって高速化を図っています。

○結果は dst_mem_no 内に－１２８～１２７の範囲で格納されます。

Lib_zero_cross

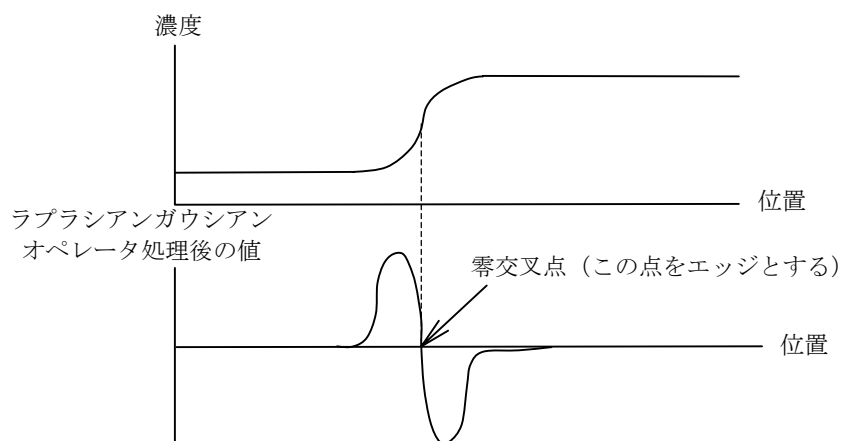
機　能　　ゼロクロッシングオペレータ

形　式

```
#include  "f_filter.h"
int Lib_zero_cross (int src_mem_no, int dst_mem_no, int neib, int val);
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

指定ウインドウ内のラプラシアンガウシアン処理後の±4ビットの深さを持つ濃淡画像において、３×３のコンボルバーを使い、零交叉点をエッジとするエッジ抽出法です。

① ***src_mem_no*** は、処理対象メモリ番号です。

② ***dst_mem_no*** は、結果格納メモリ番号です。
　src_mem_no とは、異なるメモリ番号を指定してください。

③ ***neib*** は、４近傍(=4) または ８近傍(=8)です。

④ ***val*** は、当該画素がエッジであった際に格納する値です。

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

処理後、モニタＴＶ上に表示します。

```
#include  "f_gui.h"
#include  "f_video.h"
#include  "f_pinf.h"
#include  "f_stdlib.h"
#include  "f_filter.h"

#define SIGMA 2.5
#define NEIB  8
#define VAL   255
#define SIZE  3

void disp()
{
    int dst_mem_no1;
    int dst_mem_no2;
    int status;
    int *op;
    int mult;
    int cur_mem;

    Lib_init_cursor();
    if( ( int *)NULL !=
             ( op = ( int *)Lib_mlalloc( sizeof( int ) * SIZE * SIZE )))
    {
        Lib_get_convolver( SIZE, SIGMA, op, &mult);
        if ( ERROR_RETURN != (dst_mem_no1 = Lib_alloc_gray_memory()) )
        {
            if ( ERROR_RETURN != (dst_mem_no2 = Lib_alloc_gray_memory()) )
            {
                Lib_freeze( TRANSMIT );
                status = Lib_lg_filter( (cur_mem = Lib_get_gray_memory()),
                                         dst_mem_no1, op, mult SIZE);
                if ( NORMAL_RETURN == status )
                {
                    status = Lib_zero_cross( dst_mem_no1, dst_mem_no2,
                                             NEIB, VAL);
                    if ( NORMAL_RETURN == status )
                    {
                        Lib_xvideo_transmit( dst_mem_no2, GRAY_PLANE );
                        Lib_display_keyinput( 10, 450, "入力待ち");
                    }
                }
                Lib_freerun();
            }
        }
        Lib_change_gray_memory( cur_mem );
        Lib_free_gray_memory( dst_mem_no2 );
        Lib_free_gray_memory( dst_mem_no1 );
    }
    Lib_lfree( ( char *)op);
}
```

留意事項

○処理範囲はステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、使用します。
また、それぞれ指定ウィンドウの外周の１画素は処理されません。

○処理対象メモリの濃度値の範囲は－１２８から１２７とします。

○結果は dst_mem_no 内に、０（エッジではない画素値）または val（エッジである画素値）として格納されます。

Lib_any_cross

機　能

任意値 クロッシングオペレータ

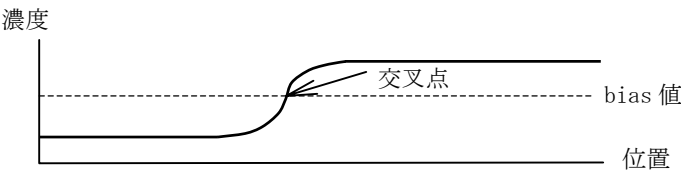
形　式

```
#include "f_filter.h"
```

int Lib_any_cross (int ***src_mem_no***, int ***dst_mem_no***, int ***neib***, int ***bias***, int ***val***);

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

説

指定ウインドウ内の８ビットの深さを持つ濃淡画像において、３×３のコンボルバーを使い、
bias 値の交叉点をエッジとするエッジ抽出法です。

① ***src_mem_no*** は、処理対象メモリ番号です。

② ***dst_mem_no*** は、結果格納メモリ番号です。
src_mem_no とは、異なるメモリ番号を指定してください。

③ ***neib*** は、４近傍(=4) または ８近傍(=8)です。

④ ***bias*** は、交叉用の基準値（上図)です。

⑤ ***val*** は、当該画素がエッジであった際に格納する値です。

戻り値

処理結果

値	定　数	意　味
0	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

処理後、モニタＴＶ上に表示します。

```
#include  "f_gui.h"
#include  "f_video.h"
#include  "f_pinf.h"
#include  "f_stdlib.h"
#include  "f_filter.h"

#define NEIB  8
#define BIAS  0
#define VAL   255

void disp()
{
    int dst_mem_no;
    int status;

    status = NORMAL_RETURN;
    Lib_init_cursor();
    if( ERROR_RETURN != ( dst_mem_no = Lib_alloc_gray_memory()) )
    {
        Lib_freeze( TRANSMIT );
        status = Lib_any_cross(Lib_get_gray_memory(), dst_mem_no,
                                                  NEIB, BIAS, VAL);
        if( NORMAL_RETURN == status)
          Lib_xvideo_trnsmit( dst_mem_no, GRAY_PLANE);
        Lib_display_key_input( 10, 450, "入力待ち");
        Lib_freerun();
    }
}
```

留意事項

○処理範囲はステージパラメータ（システムパラメータ）の「処理ウィンドウ」を参照し、使用します。
また、それぞれ指定ウィンドウの外周の１画素は処理されません。

○処理対象メモリの濃度値の範囲は０から２５５とします。

○結果は dst_mem_no 内に、０（エッジではない画素値）または val（エッジである画素値）として格納されます。

9．メモリ間転送・演算ライブラリ

本ライブラリは、濃淡画像メモリ間で、転送あるいは演算を行うものです。
本ライブラリを使用する場合、あらかじめ濃淡メモリを２面、あるいは３面確保していなければいけません。

Lib_gray_memory_move

機　能

濃淡画像転送

形　式

```
#define "f_gray.h"
int Lib_gray_memory_move( int src_mem_no, int dst_mem_no,
                                        unsigned int  mask );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

ソース・メモリ番号で指定したメモリデータ８ビットを、デスティネーション・メモリ番号で指定したメモリへ転送します。
転送時には、１画素分８ビットのデータを転送する事も、８ビットのうちの１部のビットについてのみ転送する事もできます。（「転送マスクパタン」の指定による）

① ***src_mem_no*** は、ソース（転送元）メモリ番号（０～　）です。

② ***dst_mem_no*** は、デスティネーション（転送先）メモリ番号（０～　）です。

③ ***mask*** は、どのビット位置のデータを転送するかを、ＬＳＢ　８ビットのビットパタンで指定します。（ビット値が“１”に対応するビット位置のデータが転送されます。）

戻り値

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

ＭＳＢ　４ビットを転送後、ＣＲＴ上に表示します。

```
#include "f_gray.h"
#include "f_video.h"

#define  SRC_MEM_NO     0
#define  DST_MEM_NO     1
#define  MASK_PTN       0xf0

void move_disp()
{
    Lib_freeze( TRANSMIT );
    Lib_gray_memory_move( SRC_MEM_NO, DST_MEM_NO, MASK_PTN );
    Lib_xvideo_transmit( DST_MEM_NO, GRAY_PLANE );
}
```

留意事項

○処理範囲
　処理は全面を対象に行います。

Lib_gray_memory_add

機　能

濃淡画像加算

形　式

```
#define "f_gray.h"
int Lib_gray_memory_add( int  src_mem_no1, int src_mem_no2, int  dst_mem_no );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

ソース１・メモリ番号で指定したメモリデータ８ビットと、ソース２・メモリ番号で指定したメモリデータ８ビットとを加算して、デスティネーション・メモリ番号で指定したメモリへ格納します。

① ***src_mem_no1*** は、ソース１メモリ番号（０～　）です。

② ***src_mem_no2*** は、ソース２メモリ番号（０～　）です。

③ ***dst_mem_no*** は、デスティネーション（結果格納）メモリ番号（０～　）です。
　src_mem_no1 または src_mem_no2 と同じメモリ番号でもかまいません。

戻り値

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

加算後、ＣＲＴ上に表示します。

```
#include "f_gray.h"
#include "f_video.h"

#define  SRC_MEM_NO1    0
#define  SRC_MEM_NO2    1
#define  DST_MEM_NO     2

void add_disp()
{
    Lib_freeze( TRANSMIT );
    Lib_gray_memory_add( SRC_MEM_NO1, SRC_MEM_NO2, DST_MEM_NO );
    Lib_xvideo_transmit( DST_MEM_NO, GRAY_PLANE );
}
```

留意事項

○加算結果のオーバフロー
　加算結果が２５５を越えた場合には２５５がセットされます。

○処理範囲
　処理は全面を対象に行います。

Lib_gray_memory_sub

機　能

濃淡画像減算

形　式

```
#define "f_gray.h"
int Lib_gray_memory_sub( int  src_mem_no1, int src_mem_no2, int  dst_mem_no );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

ソース１・メモリ番号で指定したメモリデータ８ビットから、ソース２・メモリ番号で指定したメモリデータ８ビットを減算して、デスティネーション・メモリ番号で指定したメモリへ格納します。

① ***src_mem_no1*** は、ソース１メモリ番号（０～　）です。

② ***src_mem_no2*** は、ソース２メモリ番号（０～　）です。

③ ***dst_mem_no*** は、デスティネーション（結果格納）メモリ番号（０～　）です。
　src_mem_no1 または src_mem_no2 と同じメモリ番号でもかまいません。

戻り値

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

減算後、ＣＲＴ上に表示します。

```
#include "f_gray.h"
#include "f_video.h"

#define  SRC_MEM_NO1    0
#define  SRC_MEM_NO2    1
#define  DST_MEM_NO     2

void sub_disp()
{
    Lib_freeze( TRANSMIT );
    Lib_gray_memory_sub( SRC_MEM_NO1, SRC_MEM_NO2, DST_MEM_NO );
    Lib_xvideo_transmit( DST_MEM_NO, GRAY_PLANE );
}
```

留意事項

○減算結果のアンダーフロー
　減算結果が０未満の場合には０がセットされます。

○処理範囲
　処理は全面を対象に行います。

10．濃度変換ライブラリ

本ライブラリは、濃淡画像メモリの濃度変換を行うものです。

Lib_binary_convert

機　能

２値化

形　式

```
#include "f_gray.h"
int Lib_binary_convert( int  bin_mem_no, int gray_mem_no, int  binary_level );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

濃淡画像データを指定の閾値により、２値化するものです。
閾値未満を黒、以上を白とします。濃淡、２値メモリＮｏ．はパラメータで指定します。

① ***bin_mem_no*** は２値画像メモリＮｏ．です。

② ***gray_mem_no*** は濃淡画像メモリＮｏ．です。

③ ***binary_level*** は２値化レベルです。０から２５５の範囲で指定してください。

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

エンハンス変換テーブルを作成し、階調変換した濃淡画像メモリを２値化します。

```
#include  "f_gui.h"
#include  "f_video.h"
#include  "f_pinf.h"
#include  "f_stdlib.h"
#include  "f_gray.h"


int gbconv()
{
    int dst_mem_no;
    int bmem_no;
    int status;
    static struct Gray_convgconv;

    Lib_display_control( GRAY_PLANE|BIN_PLANE|LINE_PLANE|CHAR_PLANE );
    status = ERROR_RETURN;
    bmem_no = Lib_get_bin_memory();
    gconv.Shape  = SLOPE_TYPE;
    gconv.Aparam = 25;
    gconv.Bparam = 25;
    Lib_init_cursor();
    if( NORMAL_RETURN == Lib_make_grayconv_table( &gconv, ON ))
    {
        if( ERROR_RETURN != (dst_mem_no = Lib_alloc_gray_memory()) )
        {
            Lib_freeze( TRANSMIT );
            status = Lib_gray_convert( Lib_get_gray_memory(), dst_mem_no );
            if( NORMAL_RETURN == status )
            {
                Lib_xvideo_transmit( dst_mem_no, GRAY_PLANE );
                Lib_display_keyinput( 10,450, "階調変換");
                status = Lib_binary_convert( bmem_no, dst_mem_no, 128 );
                if( NORMAL_RETURN == status )
                {
                    Lib_display_control( BIN_PLANE|LINE_PLANE|CHAR_PLANE );
                    Lib_xvideo_transmit( bmem_no, BIN_PLANE );
                    Lib_display_keyinput( 10,450, " 2値化 ");
                }
            }
            Lib_display_control( GRAY_PLANE|LINE_PLANE|CHAR_PLANE );
            Lib_freerun();
        }
    }
    return( status );
}
```

留意事項

ありません。

Lib_xbinary_convert

機　能

２値化Ⅱ

形　式

```
#include "f_gray.h"
int Lib_xbinary_convert( int bin_mem_no, int gray_mem_no, int binary_level );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

濃淡画像データの処理ウインド範囲を指定の２値化レベルで２値化します。
２値化レベル未満を黒、以上を白とします。
濃淡、２値メモリＮｏ．はパラメータで指定します。

① ***bin_mem_no*** は、２値画像メモリＮｏ．です。

② ***gray_mem_no*** は、濃淡画像メモリＮｏ．です。

③ ***binary_level*** は、２値化レベルです。０から２５５の範囲で指定してください。

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

始点座標（100,100）、終点座標（300,300）で囲まれた範囲を２値化します。

```
#include "f_video.h"
#include "f_gray.h"

#define      BIN_MEM_NO       0
#define      GRAY_MEM_NO      0
#define      BINARY_LEVEL     128
void binary_convert()
{
    Lib_window( 100, 100, 300, 300 );
    Lib_freeze( TRANSMIT );
    Lib_xbinary_convert( BIN_MEM_NO, GRAY_MEM_NO, BINARY_LEVEL );
}
```

留意事項

○ラインセンサモードで使用する場合は、Lib_set_extract_window にてウインド範囲を指定してください。

Lib_make_grayconv_table

機　能　　エンハンステーブルの生成

形　式

```
#include "f_gray.h"
int Lib_make_grayconv_table( struct Gray_conv *gconv, int disp );
```

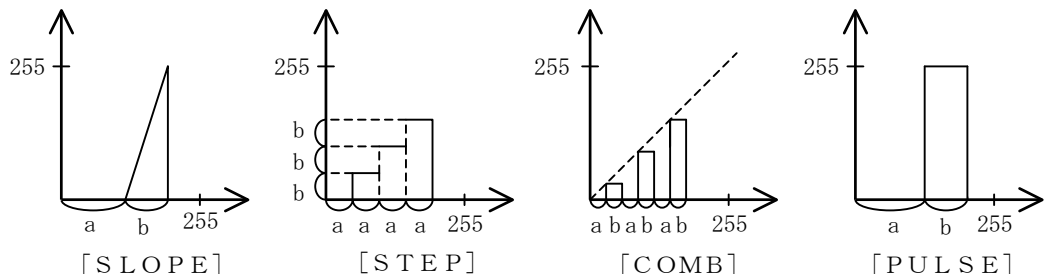
９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

設定したパラメータにより、エンハンス変換テーブルを作成します。
エンハンスの形状として４個の基本パタン（ＳＬＯＰＥ，ＳＴＥＰ，ＣＯＭＢ，ＰＵＬＳＥ）、及び１個の自由形状パタンのうちいずれか１つを指定する事ができます。
パラメータ及び、作成したエンハンス変換テーブルはシステムパラメータに更新されます。

① ***gconv*** は構造体 Gray_conv のポインタです。
パラメータは gconv が指す構造体 Gray_conv 内に格納します。
構造体 Gray_conv は f_gray.h 内で次のように宣言されています。

```
#define GRAY_LEVEL                          256

struct Gray_conv
{
    int     Shape;                          /* 形状指定              */
    int     Aparam;                         /* 形状Ａパラメータ        */
    int     Bparam;                         /* 形状Ｂパラメータ        */
    unsigned char   table[GRAY_LEVEL];      /* 自由形状指定時のテーブル*/
} ;
```

※自由形状以外を指定した場合、テーブル内は無視します。

☆ Shape はエンハンス形状です。

値	定　　数	意　　味
0	FREE_TYPE	フリー形状を指定します。
1	SLOPE_TYPE	スロープ形状を指定します。
2	STEP_TYPE	階段形状を指定します。
3	COMB_TYPE	くし型形状を指定します。
4	PULSE_TYPE	パルス形状を指定します。
5	NONE_TYPE	変換テーブルを作成しません。(LINEAR)

☆ Aparam はエンハンス形状パラメータａです。０～２５５を指定します。
エンハンス形状が FREE_TYPE, NONE_TYPE の時には意味を持ちません。

☆ Bparam はエンハンス形状パラメータｂです。０～２５５を指定します。
エンハンス形状が FREE_TYPE, NONE_TYPE の時には意味を持ちません。

☆ table はエンハンス変換テーブルです。
エンハンス形状が FREE_TYPE の場合のみ、意味を持ちます。

② ***disp*** は表示オプションです。作成したエンハンス変換テーブルをグラフ状に表示します。

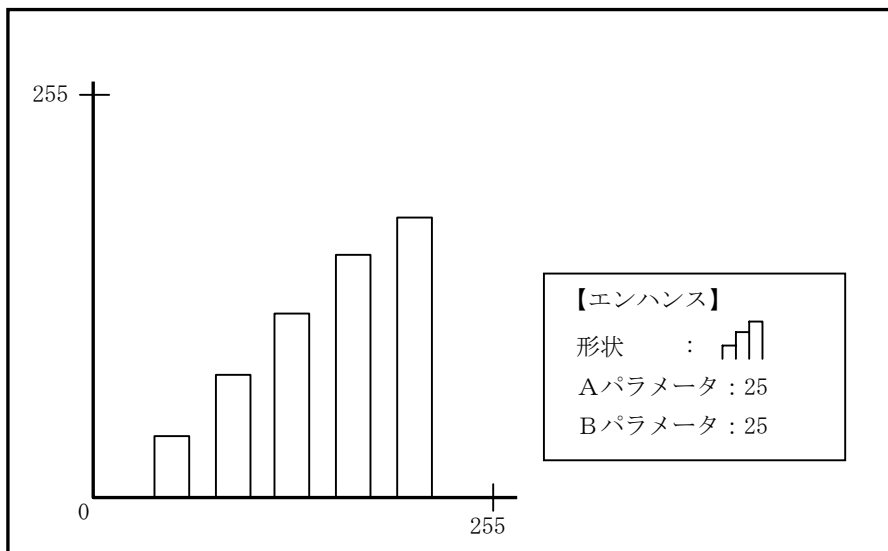
値	定　　数	意　　味
0	ＯＦＦ	表示しません。
1	ＯＮ	表示します。

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

操作方法（表示オプションをＯＮにした場合のみ）

①モニタＴＶ表示

②変換テーブル表示内容

形状（Ｓｈａｐｅ）・・・・・・・・・・・・・・・・・・エンハンス形状です。
Ａパラメータ（Ａ　ｐａｒａｍ）・・・・・・エンハンス形状パラメータａです。
Ｂパラメータ（Ｂ　ｐａｒａｍ）・・・・・・エンハンス形状パラメータｂです。

③トラックボール操作

ボタンを押す（ EXECUTE，CANCELキー押下）・・・・・・ 終了します。

例

エンハンス変換テーブルを作成し、階調変換を実施します。

```
#include  "f_gui.h"
#include  "f_video.h"
#include  "f_pinf.h"
#include  "f_stdlib.h"
#include  "f_gray.h"

int gconv()
{
    int dst_mem_no;
    int status;
    static struct Gray_convgconv;

    status = ERROR_RETURN;
    gconv.Shape  = SLOPE_TYPE;
    gconv.Aparam = 25;
    gconv.Bparam = 25;
    Lib_init_cursor();
    if( NORMAL_RETURN == Lib_make_grayconv_table( &gconv, ON ))
    {
        if( ERROR_RETURN != (dst_mem_no = Lib_alloc_gray_memory()) )
        {
            Lib_freeze( TRANSMIT );
            status = Lib_gray_convert( Lib_get_gray_memory(), dst_mem_no );
            if( NORMAL_RETURN == status )
            Lib_xvideo_transmit( dst_mem_no, GRAY_PLANE );
            Lib_display_keyinput( 10,450, "入力待ﾁ");
            Lib_freerun();
        }
    }
    return( status );
}
```

留意事項

○表示オプションをＯＮに指定した場合は、Lib_init_cursor( )を実行前にコールしておいてください。

○作成したエンハンス変換テーブルはシステムパラメータを更新しただけなので、毎回使用する場合は必ずファイル装置にシステムパラメータをセーブしてください。
セーブしていない場合は電源ＯＦＦ時に消去されてしまいます。

Lib_gray_convert

機　能

階調変換

形　式

```
#include "f_gray.h"
int Lib_gray_convert( int src_mem_no, int dst_mem_no );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

システムパラメータのエンハンス変換テーブルを使用して線形変換を行います。
処理範囲は指定ウインドウ内のみ行います。

① ***src_mem_no*** は処理対象メモリ番号です。

② ***dst_mem_no*** は結果格納メモリ番号です。
　src_mem_no と同じメモリ番号を指定してもかまいません。

戻り値

処理結果

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

エンハンス変換テーブルを作成し、階調変換を実施します。

```
#include  "f_gui.h"
#include  "f_video.h"
#include  "f_pinf.h"
#include  "f_stdlib.h"
#include  "f_gray.h"

int gconv()
{
    int dst_mem_no;
    int status;
    static struct Gray_convgconv;

    status = ERROR_RETURN;
    gconv.Shape  = SLOPE_TYPE;
    gconv.Aparam = 25;
    gconv.Bparam = 25;
    Lib_init_cursor();
    if( NORMAL_RETURN == Lib_make_grayconv_table( &gconv, ON ))
    {
        if( ERROR_RETURN != (dst_mem_no = Lib_alloc_gray_memory()) )
        {
            Lib_freeze( TRANSMIT );
            status = Lib_gray_convert( Lib_get_gray_memory(), dst_mem_no );
            if( NORMAL_RETURN == status )
            Lib_xvideo_transmit( dst_mem_no, GRAY_PLANE );
            Lib_display_keyinput( 10,450, "入力待ﾁ");
            Lib_freerun();
        }
    }
    return( status );
}
```

留意事項

ありません。

11．画像計測ライブラリ

本ライブラリは、指定された処理範囲の画像データについて濃度計測データを求めるものです。

Lib_projection

機　能

濃度投影

形　式

```
#include "f_gray.h"
```

９０Ｘ

long Lib_projection(int ***memory_no***, long ****sumx***, long ****sumy***,
　　　　　　　　　　　　　　　　　　　struct _prjinfo ****prjinfo***, int ***flag***);

FVL/LNX

int Lib_projection(int ***memory_no***, int ****sumx***, int ****sumy***,
　　　　　　　　　　　　　　　　　　　struct _prjinfo ****prjinfo***, int ***flag***);

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

カレントステージＮｏのウィンドウ（処理範囲）内のグレイメモリについて、累積濃度、累積濃度の最大位置、最小位置、平均値を横、縦方向についてもとめ、ヒストグラムを表示します。

① ***memory_no*** はメモリＮｏ．です。

② ****sumx*** は横方向累積濃度を格納する配列のポインタです。
配列の大きさは縦方向画素数分必要です。
縦方向位置を配列のインデックスとして、対応する累積濃度が格納されます。
ウィンドウ（処理範囲）外は格納されません（前の値がそのまま残っています）。
配列のポインタにＮＵＬＬが指定されると横方向の処理は行いません。

③ ****sumy*** は縦方向累積濃度を格納する配列のポインタです。
配列の大きさは横方向画素数分必要です。
横方向位置を配列のインデックスとして、対応する累積濃度が格納されます。
ウィンドウ（処理範囲）外は格納されません（前の値がそのまま残っています）。
配列のポインタにＮＵＬＬが指定されると縦方向の処理は行いません。

④ ****prjinfo*** は構造体 _prjinfo のポインタです。

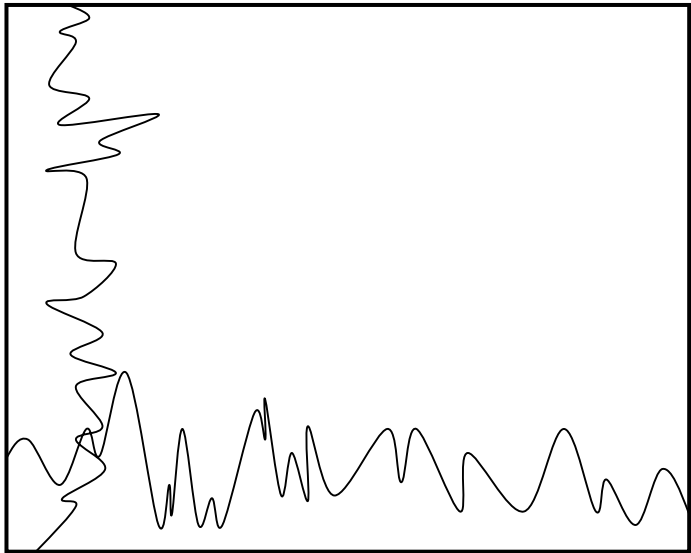
もとめた値は prjinfo が指す構造体 _prjinfo 内に格納します。
構造体 _prjinfo は f_gray.h 内で次のように宣言されています。

```
struct _prjinfo
{
    short max_xposition;    /* X(横)方向最大値座標 */
    short min_xposition;    /* X(横)方向最小値座標 */
    short max_yposition;    /* Y(縦)方向最大値座標 */
    short min_yposition;    /* Y(縦)方向最小値座標 */
    long average_x;         /* X(横)方向平均値     */    (注)
    long average_y;         /* Y(縦)方向平均値     */    (注)
};
```

(注)：FVL/LNXではint型となります

⑤ ***flag*** でヒストグラムの表示を切り替えます。

値	定　　数	意　　味
0	ＯＦＦ	表示しません。
1	ＯＮ	表示します。

戻り値

処理結果

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

メモリを確保後画像を入力し、ヒストグラムを表示します。

```
#include "f_stdlib.h"
#include "f_video.h"
#include "f_gray.h"

void projection( void )
{
    int mem_no;
    struct _prjinfo prj;
    long sumx[ 480 ];
    long sumy[ 512 ];

    if ( ERROR_RETURN != ( mem_no = Lib_alloc_gray_memory() ) )
    {
        if ( NORMAL_RETURN == Lib_change_gray_memory( mem_no ) )
        {
            Lib_input_video_control( GRAY_PLANE );
            Lib_freeze( TRANSMIT );

            Lib_projection( mem_no, sumx, sumy, &prj, 1 );

            /* 何ﾗｶﾉ処理 */

            Lib_change_gray_memory( 0 );
        }
        Lib_free_gray_memory( mem_no );
    }
}
```

留意事項

○*sumx, *sumy ともＮＵＬＬが指定された場合は異常終了となります。

Lib_stddevi

機　能

最大、最小、平均、標準偏差

形　式

```
#include "f_gray.h"
long Lib_stddevi( int memory_no, struct _deviinfo *deviinfo );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	△

解　説

カレントステージＮｏのウィンドウ（処理範囲）内のグレイメモリについて、最大、最小、平均、標準偏差をもとめます。

① ***memory_no*** はメモリＮｏ．です。

② ****deviinfo*** は構造体 _deviinfo のポインタです。

もとめた値は deviinfo が指す構造体 _deviinfo 内に格納します。
構造体 _deviinfo は f_gray.h 内で次のように宣言されています。

```
struct _deviinfo
{
    unsigned char max;          /* 最大値   */
    unsigned char min;          /* 最小値   */
    double average;             /* 平均値   */
    double stddeviation;        /* 標準偏差 */
};
```

戻り値

処理結果

値	定　数	意　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	異常終了しました。

例

メモリを確保後画像を入力し、標準偏差をもとめます。

```
#include "f_stdlib.h"
#include "f_video.h"
#include "f_gray.h"

void stddevi( void )
{
    int mem_no;
    struct _deviinfo devi;

    if ( ERROR_RETURN != ( mem_no = Lib_alloc_gray_memory() ) )
    {
        if ( NORMAL_RETURN == Lib_change_gray_memory( mem_no ) )
        {
            Lib_input_video_control( GRAY_PLANE );
            Lib_freeze( TRANSMIT );

            Lib_stddevi( mem_no, &devi );

            /* 何らかの処理 */

            Lib_change_gray_memory( 0 );
        }
        Lib_free_gray_memory( mem_no );
    }
}
```

留意事項

○画像のサイズが4000×4000画素まで対応しています。

Lib_edge_pos_xy

機　能

エッジ検出

形　式

```
#include "f_gray.h"
int Lib_edge_pos_xy( int wXs, int wYs, int wXe,
                     int wYe, int wCrossMode, int wMinMaxDiff,
                        int wMemoryNo, int *wXpos, int *wYpos );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

２点間の直線上にあるエッジを検出します。
求めたいエッジが直線上に１つである時を対象としています。
処理の方法は、まず１次微分値の最大値を仮のエッジ位置とします。次に仮のエッジ位置付近で、２次微分値がゼロクロスする位置をエッジとして検出します。

① ***wXs*** は測定開始Ｘ位置です。

② ***wYs*** は測定開始Ｙ位置です。

③ ***wXe*** は測定終了Ｘ位置です。

④ ***wYe*** は測定終了Ｙ位置です。

⑤ ***wCrossMode*** はゼロクロス位置の処理方法です。仮のエッジ付近を２次微分するときに使用するデータを指定します。１の場合は生画像のデータをそのまま使い、０の場合は平均化したデータを使用します。

値	定　　数	意　　味
1	なし	生データを指定します。
0	なし	平均データを指定します。

⑥ ***wMinMaxDiff*** は最大微分値と最小微分値の差の最小しきい値です。
（０～２５５）

⑦ ***wMemoryNo*** はエッジ検出するグレイメモリＮｏ．です。

⑧ ****wXpos*** は検出エッジ位置Ｘ座標です。（１６倍値）

⑨ ****wYpos*** は検出エッジ位置Ｙ座標です。（１６倍値）

戻り値

値	定　　数	意　　味
0	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	エッジが存在しません。 または異常終了しました。

例

点Ａ（０，０）と点Ｂ（５１１，４７９）の２点間のエッジを検出します。

```
/*ヘッダー*/
#include "f_graph.h"
#include "f_stdlib.h"
#include "f_stdio.h"
#include "f_video.h"
#include "f_image.h"
#include "f_pinf.h"
#include "f_gui.h"
#include "f_gray.h"

/*サンプルメイン*/
void main()
{
    int     wXs, wYs, wXe, wYe;
    int     wCrossMode, wMinMaxDiff;
    int     wXpos, wYpos;
    char    string[80];

    Lib_input_video_control( GRAY_PLANE );
    Lib_display_control( GRAY_PLANE | LINE_PLANE | CHAR_PLANE );
    Lib_memory_clear( LINE_PLANE | CHAR_PLANE );
    Lib_init_cursor();
    Lib_freerun();

    wXs  = 0;  wYs  = 0;  wXe  = 511;  wYe  = 479;
    wCrossMode = 1;    wMinMaxDiff = 30;

    Lib_freeze( TRANSMIT );
    if( NORMAL_RETURN != ( Lib_edge_pos_xy( wXs, wYs, wXe, wYe,
                            wCrossMode, wMinMaxDiff, CURRENT_MEMORY,
                            &wXpos, &wYpos )))
    {
        Lib_display_message( 400, 450, "エラー", "エッジ検出エラー！！");
    }
    else
    {
        Lib_pset( wXpos/16, wYpos/16, GRAPH_DRAW );
        Lib_sprintf( string, "エッジ点 (%d,%d) ", wXpos/16, wYpos/16 );
        Lib_chrdisp( 1, 1, string );
        Lib_display_message( 400, 450, "STOP", "エッジ検出点表¥示");
    }
}
```

留意事項

○２点間にエッジが２つ以上存在する場合には、１次微分値の最も大きい所がエッジ位置となります。

○第６パラメータの wMinMaxDiff（最大微分値と最小微分値の差の最小しきい値）とは、一次微分時の最大微分値と最小微分値の差を求め、その差がこの値以下だったら、エッジとして採用しないというものです。

見た目でエッジが存在しないような画像（例えば、画面全体が真っ白または真っ黒な画像など）でも、濃度値の変化は、微少ながらに存在しています。

エッジ検出では、濃度値の変化が最も大きい所をエッジ位置として採用しているわけですが、例えば、見た目に真っ白に見える画像でも、濃度値が２４５から２５５までなどと変化していたりします。

ですから、このような場合でも、変化が最大の所を無理矢理エッジとして検出してしまいます。

このパラメータがあることで、例えば、wMinMaxDiff の値を３０と設定すると、上の例の場合はエッジがないと判断してライブラリはー１を返します。

つまり、このパラメータが有ることで ”エッジが存在しない” という認識も可能になります。

Lib_edge_pos_xy2

機　能　　エッジ検出２

形　式

```
#include "f_gray.h"
int Lib_edge_pos_xy2( int wXs, int wYs, int wXe, int wYe, int wCrossMode,
                      int wMinMaxDiff, int wMemoryNo, int wDetectMode,
                      int *wXpos, int *wYpos );
```

９０１	９０２	９０３	９０４	FVL/LNX	高分解能
○	○	○	○	○	○

解　説

２点間の直線上にあるエッジを検出します。
求めたいエッジが直線上に１つである時を対象としています。
処理の方法は、まず１次微分値の最大値を仮のエッジ位置とします。次に仮のエッジ位置付近で、２次微分値がゼロクロスする位置をエッジとして検出します。

① ***wXs*** は測定開始Ｘ位置です。

② ***wYs*** は測定開始Ｙ位置です。

③ ***wXe*** は測定終了Ｘ位置です。

④ ***wYe*** は測定終了Ｙ位置です。

⑤ ***wCrossMode*** はゼロクロス位置の処理方法です。仮のエッジ付近を２次微分するときに使用するデータを指定します。１の場合は生画像のデータをそのまま使い、０の場合は平均化したデータを使用します。

値	定　　数	意　　味
1	なし	生データを指定します。
0	なし	平均データを指定します。

⑥ ***wMinMaxDiff*** は最大微分値と最小微分値の差の最小しきい値です。
（０～２５５）

⑦ ***wMemoryNo*** はエッジ検出するグレイメモリＮｏ．です。

⑧ ***wDetectMode*** はエッジ検出モードです。

値	定　　数	意　　味
0	MAX_DIFF_POINT	検出ライン上の最大微分値の位置をエッジ位置とします。
1	DARK_LIGHT_POINT	検出ライン上の暗から明に変わる位置をエッジ位置とします。
2	LIGHT_DARK_POINT	検出ライン上の明から暗に変わる位置をエッジ位置とします。

⑨ ***wXpos** は検出エッジ位置Ｘ座標です。（１６倍値）

⑩ ***wYpos** は検出エッジ位置Ｙ座標です。（１６倍値）

戻り値

値	定　　数	意　　味
０	NORMAL_RETURN	正常終了しました。
－１	ERROR_RETURN	エッジが存在しません。 または異常終了しました。

例

点Ａ（０，０）と点Ｂ（５１１，４７９）の２点間のエッジを検出します。

```
/*ヘッダー*/
#include "f_graph.h"
#include "f_stdlib.h"
#include "f_stdio.h"
#include "f_video.h"
#include "f_image.h"
#include "f_pinf.h"
#include "f_gui.h"
#include "f_gray.h"

/*サンプルメイン*/
void main()
{
    int     wXs, wYs, wXe, wYe;
    int     wCrossMode, wMinMaxDiff, wDetectMode;
    int     wXpos, wYpos;
    char    string[80];

    Lib_input_video_control( GRAY_PLANE );
    Lib_display_control( GRAY_PLANE | LINE_PLANE | CHAR_PLANE );
    Lib_memory_clear( LINE_PLANE | CHAR_PLANE );
    Lib_init_cursor();
    Lib_freerun();

    wXs  = 0;  wYs  = 0;  wXe  = 511;  wYe  = 479;
    wCrossMode = 1;    wMinMaxDiff = 30;  wDetectMode = MAX_DIFF_POINT;

    Lib_freeze( TRANSMIT );
    if( NORMAL_RETURN != ( Lib_edge_pos_xy2( wXs, wYs, wXe, wYe,
                           wCrossMode, wMinMaxDiff, CURRENT_MEMORY,
                           wDetectMode, &wXpos, &wYpos )))
    {
        Lib_display_message( 400, 450, "エラー", "エッジ検出エラー！！");
    }
    else
    {
        Lib_pset( wXpos/16, wYpos/16, GRAPH_DRAW );
        Lib_sprintf( string, "エッジ点 (%d,%d) ", wXpos/16, wYpos/16 );
        Lib_chrdisp( 1, 1, string );
        Lib_display_message( 400, 450, "STOP", "エッジ検出点表¥示");
    }
}
```

留意事項

○２点間にエッジが２つ以上存在する場合には、１次微分値の最も大きい所がエッジ位置となります。
検出モードが０の場合は最大微分値の位置、検出モードが１の場合は暗から明へ変化する位置に限定した最大微分値の位置、検出モードが２の場合は明から暗へ変化する位置に限定した最大微分値の位置となります。

○第６パラメータの wMinMaxDiff（最大微分値と最小微分値の差の最小しきい値）とは、一次微分時の最大微分値と最小微分値の差を求め、その差がこの値以下だったら、エッジとして採用しないというものです。
見た目でエッジが存在しないような画像（例えば、画面全体が真っ白または真っ黒な画像など）でも、濃度値の変化は、微少ながらに存在しています。
エッジ検出では、濃度値の変化が最も大きい所をエッジ位置として採用しているわけですが、例えば、見た目に真っ白に見える画像でも、濃度値が２４５から２５５までなどと変化していたりします。
ですから、このような場合でも、変化が最大の所を無理矢理エッジとして検出してしまいます。
このパラメータがあることで、例えば、wMinMaxDiff の値を３０と設定すると、上の例の場合はエッジがないと判断してライブラリは－１を返します。
つまり、このパラメータが有ることで ”エッジが存在しない” という認識も可能になります。

Lib_edge_pos_xy，Lib_edge_pos_xy2 のｱﾙｺﾞﾘｽﾞﾑについて

１．Lib_edge_pos_xy、Lib_edge_pos_xy2におけるエッジ検出の前提条件

「濃淡エッジ計測ライブラリ」は、計測ライン上の微分しきい値を越えた場所すべてをエッジ位置として検出します。

それに対して Lib_edge_pos_xy，Lib_edge_pos_xy2 は、計測ライン上の最大微分値の場所をエッジ位置として１点検出します。

「濃淡エッジ計測ライブラリ」では検出できなかった図１，図２のような計測ライン上に１点のみエッジが存在する場合でもエッジを検出できます。

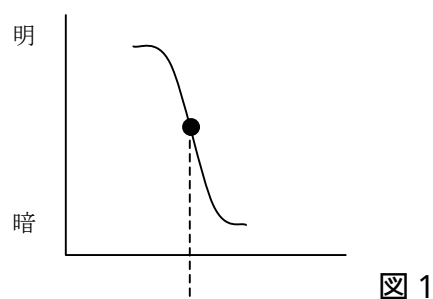
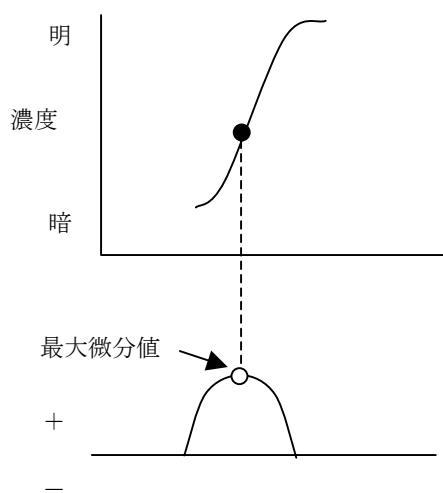



図１

図２

２．エッジ計測のアルゴリズムについて

計測ライン上の濃度波形が図３のような場合を考えます。まず、計測ライン上の濃度値を微分します。求めた微分値をプロットすると図４のような波形になります。

次に求めた微分値から最大微分値と最小微分値の差を求め、最大微分値と最小微分値の差のしきい値以上であるかを判断します(次項を参照)。

しきい値を越えていれば最大微分値の位置を仮のエッジ点(図４の白丸)とします。

次に、仮のエッジ位置近辺の濃度値を２次微分しゼロクロスする位置(図５の白丸)のエッジ座標を検出します。

このとき、ゼロクロス位置の処理方法(wCrossMode)に「平均データ」を指定した場合、仮のエッジ位置近辺の濃度値をあらかじめ平均化した後、２次微分を行います。

計測ライン上にノイズ等があり検出結果が安定しない場合などに平均データを指定すると改善される可能性があります。

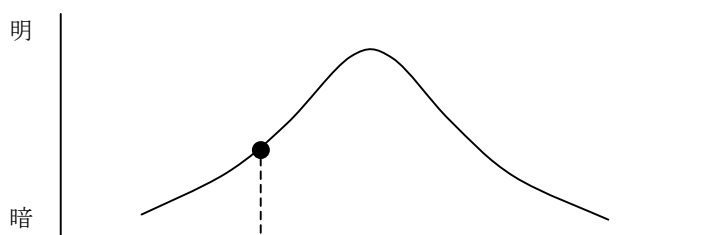



図３　濃度波形

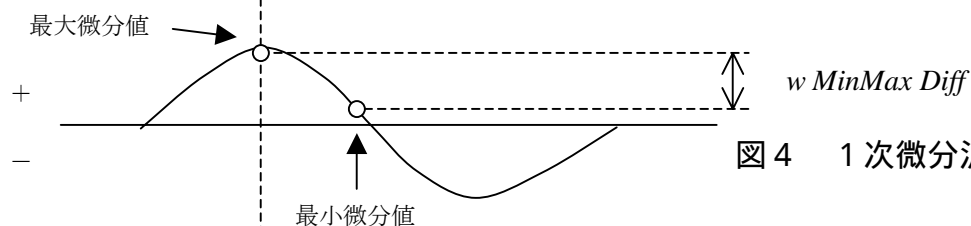



図４　１次微分波形

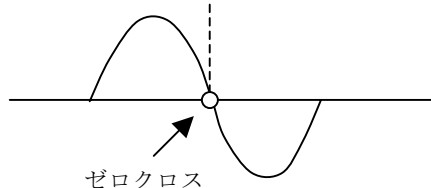



図５　２次微分波形

３．最大微分値と最小微分値の差のしきい値(wMinMaxDiff)について

図６のように、計測ライン上の濃度変化が非常に小さく人間が画像を見た場合エッジはないと判断するような画像の場合を考えます。

しかしながら、このような場合においても微少ながらも濃度変化は存在し、その変化が最大である最大微分値の箇所は必ず存在します。

最大微分値の位置をエッジとして検出するので、図６のような画像でも最大微分値の位置がエッジとして検出されてしまいます。

そこで、最大微分値と最小微分値の差のしきい値を設定する事により、計測ライン上の最大微分値と最小微分値の差を求め、その値がしきい値以下であれば、計測ライン上にエッジは存在しないという判断をすることができます。

図６　微小な変化でも検出

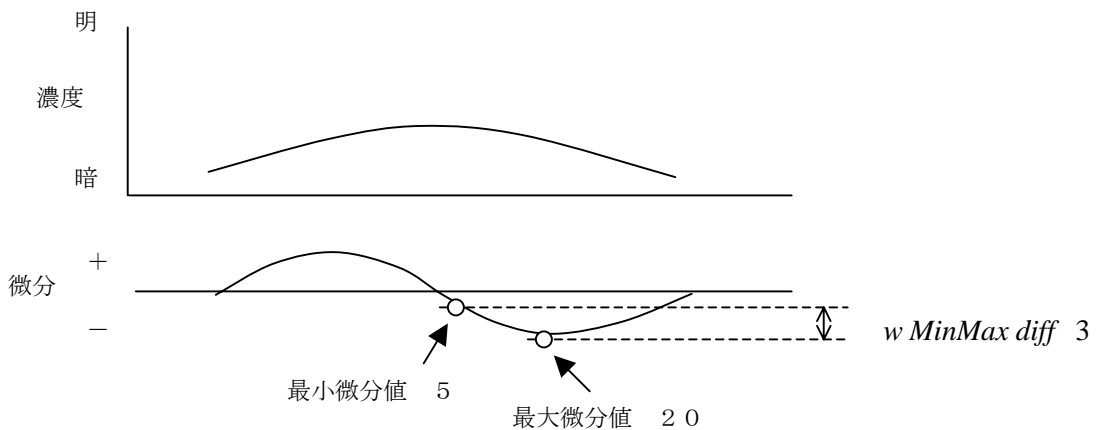
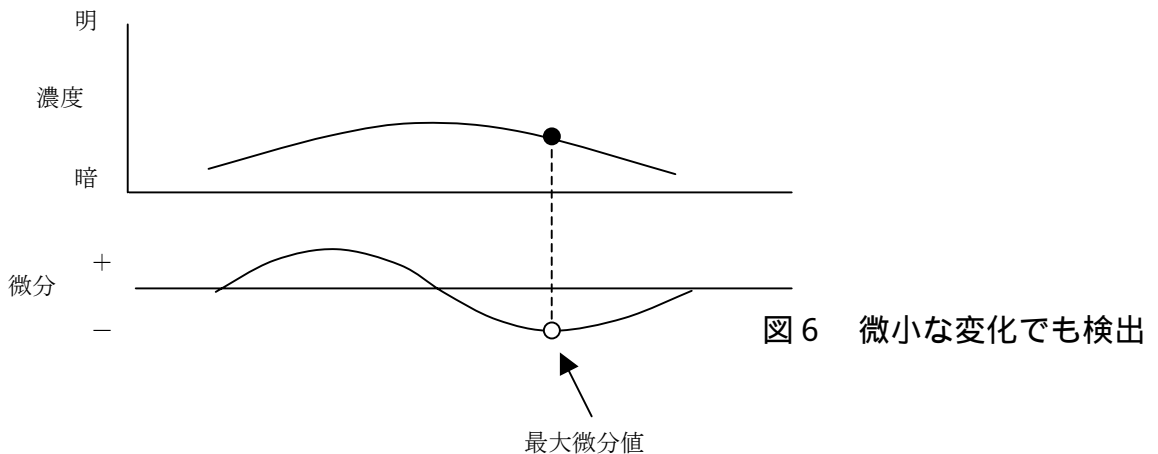



図７　エッジは存在しない

例）　２０　－　５　<　３０　なのでエッジは存在しない

４．エッジ検出モード(wDetectMode)について

Lib_edge_pos_xy2 ではエッジ検出モードを指定できます。
MAX_DIFF_POINT を指定すると計測ライン上の最大微分値の位置をエッジとして検出します。
DARK_LIGHT_POINT を指定すると図８のように計測ライン上の暗から明に変わる最大微分値の位置をエッジとして検出します。
LIGHT_DARK_POINT を指定すると図９のように計測ライン上の明から暗に変わる最大微分値の位置をエッジとして検出します。

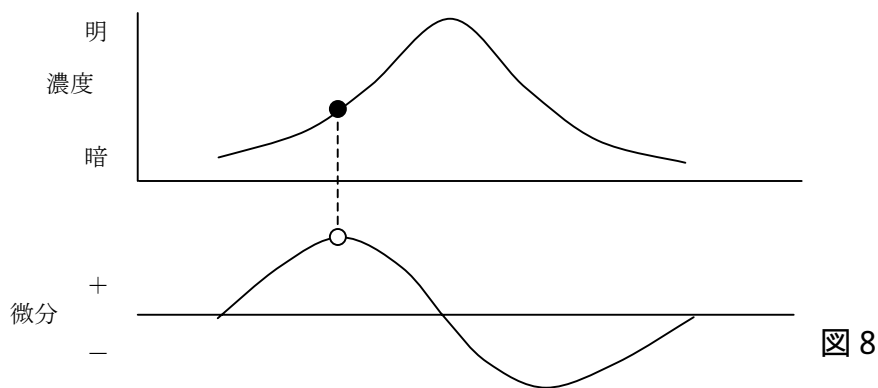



図８

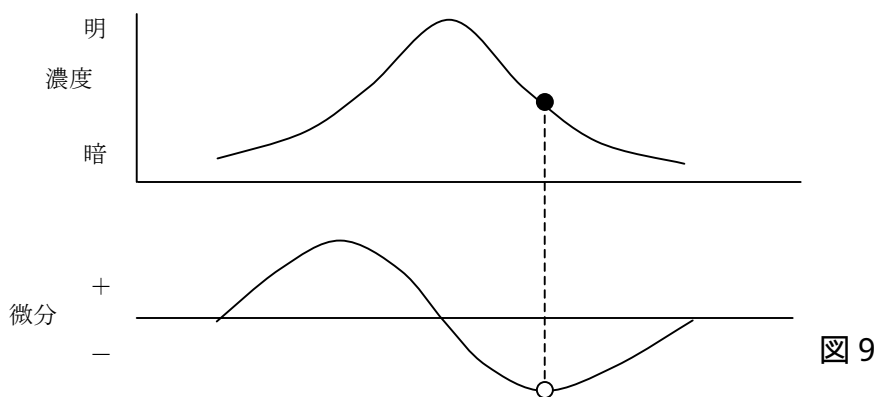



図９

付録１．各ライブラリの処理速度一覧

○計測対象機種

９０１	**ＣＳＣ９０１ＮＴ**
９０２	**ＦＶ９０２　ｍｏｄｅｌ－３** (Celeron 433MHz)
	S回転サーチはＣＳＣ９０２ＳＴ－ＲＩＣＥ (MMX Pentium200MHz)
９０３	**ＣＳＣ９０３**
９０４	**ＦＶ９０４**
FVL/LNX	**ＦＶ２０００**

各ライブラリの実行所要時間欄の

空　白 ➡ 使用不可

――― ➡ 条件により変化

・・・・・・ ➡ 測定不能（所要時間が0.001ミリ秒未満）

を表しています。

濃淡画像ライブラリ実行所要時間代表値一覧

単位　　ミリ秒

項番	ライブラリ名	所要時間					備　　考
		９０１	９０２	９０３	９０４	FVL/LNX	
1	Lib_any_cross	474.445	21.159	225.504	93.587	9.700	８近傍
2	Lib_averaging	109.863	8.138	68.034	19.531	3.300	[*1]
3	Lib_binary_convert	42.318	4.069	38.249	11.393	3.000	
4	Lib_edge_pos_xy	1.628	0.000	1.221	0.244	0.000	(73,115) –(435,115)
5	Lib_edge_pos_xy2	0.977	0.000	1.221	0.244	0.000	(73,115) –(435,115)
6	Lib_em_avr_inspection	2.441	0.130	1.953	0.814	0.090	
7	Lib_em_avr_inspection2	3.255	0.130	2.441	0.814	0.090	
8	Lib_em_calib	0.002	・・・・・・	0.002	0.002	・・・・・・	
9	Lib_em_edge_disp	―――	―――	―――		―――	
10	Lib_em_edge_pos	2.441	0.130	1.953	0.814	0.090	
11	Lib_em_edge_pos2	2.441	0.130	2.035	0.814	0.090	
12	Lib_em_inspection	2.604	0.146	2.441	0.814	0.090	
13	Lib_em_inspection_close	0.760	0.041	1.600	0.217	0.060	open + close
14	Lib_em_inspection_open	0.760	0.041	1.600	0.217	0.060	open + close
15	Lib_em_inspection2	3.255	0.146	2.441	0.814	0.090	
16	Lib_es_calculation	906.573	102.539	697.020	212.402	30.000	[*2]
17	Lib_es_change_dictionary_size	―――	―――	―――	―――	―――	

[*1]ＭＭＸ対応（システム Ver1.60以降）です。
[*2]パタンサイズ　64×64　　pick_n……1　　rand_n……1　　サーチエリア　512×480

濃淡画像ライブラリ実行所要時間代表値一覧

単位　　ミリ秒

項番	ライブラリ名	所要時間					備　　考
		９０１	９０２	９０３	９０４	FVL/LNX	
18	Lib_es_del_pattern	―――	―――	―――	―――	―――	
19	Lib_es_error_message	―――	―――	―――	―――	―――	
20	Lib_es_get_dictionary_size	―――	―――	―――	―――	―――	
21	Lib_es_get_pattern_n	―――	―――	―――	―――	―――	
22	Lib_es_get_pattern_name	―――	―――	―――	―――	―――	
23	Lib_es_init_dictionary	―――	―――	―――	―――	―――	
24	Lib_es_reg_pattern	―――	―――	―――	―――	―――	
25	Lib_es_set_max_edge	―――	―――	―――	―――	―――	
26	Lib_fdefferential	130.208	8.138	87.239	23.600	3.300	Ｘ方向[*1]
		98.470	8.138	80.078	23.600	3.100	Ｙ方向[*1]
		215.331	8.138	120.117	48.828	4.500	ＸＹ方向[*1]
27	Lib_get_convolver	8.138	0.163	20.996	0.814	0.000	SIZE 9
28	Lib_gray_convert	56.152	8.138	36.621	11.393	3.300	
29	Lib_gray_memory_add	73.242	9.766	91.552	43.945	2.500	[*1]
30	Lib_gray_memory_move	21.973	6.510	20.508	5.697	2.000	[*1]
31	Lib_gray_memory_sub	60.221	9.766	62.093	33.366	2.500	[*1]
32	Lib_gs_1dsppat	―――	―――	―――	―――	―――	
33	Lib_gs_adjmark	―――	―――	―――	―――	―――	
34	Lib_gs_close_data_file	―――	―――	―――	―――	―――	
35	Lib_gs_defadrs	0.20	0.000	0.081	······	······	

[*1]ＭＭＸ対応（システム Ver1.60以降）です。

濃淡画像ライブラリ実行所要時間代表値一覧

単位　ミリ秒

項番	ライブラリ名	所要時間					備　考
		９０１	９０２	９０３	９０４	FVL/LNX	
36	Lib_gs_defmask	―――	―――	―――	―――	―――	
37	Lib_gs_defpat	―――	―――	―――	―――	―――	
38	Lib_gs_disp_result	―――	―――	―――	―――	―――	
39	Lib_gs_dsppat	―――	―――	―――	―――	―――	
40	Lib_gs_exdefpat	―――	―――	―――	―――	―――	
41	Lib_gs_fremask	2.94	2.441	35.807	10.579	1.000	パタンサイズ　32×32
42	Lib_gs_get_data_file_adrs	0.02	0.001	0.000	0.000	・・・・・・	
43	Lib_gs_get_ptnname					・・・・・・	
44	Lib_gs_get_ptnnum					・・・・・・	
45	Lib_gs_gfreeze	注１	注１	注１	注１	注１	注１：P295 または P298
46	Lib_gs_infpat	―――	―――	―――	―――	―――	
47	Lib_gs_open					―――	
48	Lib_gs_open_data_file	―――	―――	―――	―――	―――	
49	Lib_gs_pcorr	注１	注１	注１	注１	注１	注１：P295 または P298
50	Lib_gs_point_search	443	16.276	645	119	8.000	パタン　32×32 ウインドウ　64×64
51	Lib_gs_psearch	―――	―――	―――	―――	―――	
52	Lib_gs_ptn_compare	注１	注１	注１	注１	注１	注１：P295 または P298
53	Lib_gs_ptn_get	―――	―――	―――	―――	―――	
54	Lib_gs_save_data_file	―――	―――	―――	―――	―――	

［*1］MMX対応（システム Ver1.60以降）です。

濃淡画像ライブラリ実行所要時間代表値一覧

単位　ミリ秒

項番	ライブラリ名	所要時間					備　考
		９０１	９０２	９０３	９０４	FVL/LNX	
55	Lib_gs_scondition	0.02	0.000	0.000	0.000	・・・・・・	
56	Lib_gs_search	注２	注２	注２	注２	注２	注２：P291 または P296
57	Lib_gs_smode	0.02	0.000	0.000	0.000	・・・・・・	
58	Lib_gs_u1param	0.02	0.000	0.000	0.000	・・・・・・	
59	Lib_gs_upmark	0.02	0.000	0.000	0.000	・・・・・・	
60	Lib_gs_usedel	0.27	0.000	0.000	4.069	・・・・・・	
61	Lib_gs_usepat	22	1.628	13.835	4.069	2.100	128×128
62	Lib_gs_usepat_circle	700	83.008	446.776	257.161	137.000	半径６０
63	Lib_gs_usepat_square	138	21.973	89.518	26.855	7.000	(100,100)-(300,300)30%
64	Lib_gs_window	0.02	0.000	0.000	0.000	0.000	白枠表示なし
65	Lib_gs_xcondition	0.02	0.000	0.000	0.000	0.000	
66	Lib_gs_xdefadrs	0.20	0.000	0.081	0.081	0.000	
67	Lib_gs_xsearch	注３	注３	注３	注３	注３	注３：P293 または P297
68	Lib_gs_ycondition	0.02	0.000	0.000	0.000	0.000	
69	Lib_gs_ysearch	注３	注３	注３	注３	注３	注３：P293 または P297 さらに回転の為の処理時間が必要になります。
70	Lib_laplacian	141.601	8.138	107.584	30.924	3.400	[*1]
71	Lib_lg_filter	9536.108	348.306	8898.089	3359.366	220.000	size 9
72	Lib_lhough_close	―――	―――	―――	―――	―――	
73	Lib_lhough_detection	24.414	3.255	26.081	8.138	1.000	

[*1]ＭＭＸ対応（システム Ver1.60以降）です。

濃淡画像ライブラリ実行所要時間代表値一覧

単位　　ミリ秒

項番	ライブラリ名	所要時間					備　考
		９０１	９０２	９０３	９０４	FVL/LNX	
74	Lib_lhough_open	―――	―――	―――	―――	―――	
75	Lib_lhough_voting	―――	―――	―――	―――	―――	
76	Lib_make_grayconv_table	0.222	0.010	0.102	0.039	0.006	
77	Lib_max_filter	152.994	8.138	96.354	26.042	3.600	[*1]
78	Lib_max4_filter	120.442	8.138	120.442	39.062	3.500	ＸＹ方向[*1] mode0
79	Lib_median	171.712	8.138	106.933	30.111	3.500	[*1]
80	Lib_min_filter	153.808	8.138	96.354	26.042	3.300	[*1]
81	Lib_min4_filter	120.442	8.138	120.442	39.062	3.300	ＸＹ方向[*1] mode0
82	Lib_projection	103.353	10.579	159.342	60.221	4.000	ＸＹ方向[*1]
83	Lib_Roberts	112.304	8.138	95.215	21.159	3.100	[*1]
84	Lib_sdefferential	138.346	8.138	107.259	31.738	3.800	[*1]
85	Lib_sharp	140.787	8.138	119.466	32.552	3.800	[*1]
86	Lib_sobel	131.836	8.138	87.565	24.414	3.400	Ｘ方向[*1]
		102.539	8.138	87.890	24.414	3.300	Ｙ方向[*1]
		232.747	8.138	136.067	34.993	5.200	ＸＹ方向[*1]
87	Lib_srs_close		······			······	
88	Lib_srs_get_fine_srch_sw		······			······	
89	Lib_srs_get_mask_ptn		2			0.200	
90	Lib_srs_get_ptn_image		4			0.200	ﾊﾟﾀﾝｻｲｽﾞ　250×250
91	Lib_srs_get_ptn_image_size		······			······	

[*1]ＭＭＸ対応（システム Ver1.60以降）です。

濃淡画像ライブラリ実行所要時間代表値一覧

単位　　ミリ秒

項番	ライブラリ名	所要時間					備　　考
		９０１	９０２	９０３	９０４	FVL/LNX	
92	Lib_srs_get_ptn_param		・・・・・・			・・・・・・	
93	Lib_srs_get_rgst_ptn_names		・・・・・・			・・・・・・	
94	Lib_srs_get_rgst_ptn_num		・・・・・・			・・・・・・	
95	Lib_srs_get_speed		・・・・・・			・・・・・・	
96	Lib_srs_mask_define		・・・・・・			・・・・・・	
97	Lib_srs_open		・・・・・・			・・・・・・	
98	Lib_srs_ptn_close		・・・・・・			・・・・・・	
99	Lib_srs_ptn_delete		・・・・・・			・・・・・・	
100	Lib_srs_ptn_load		―――			―――	
101	Lib_srs_ptn_modify		・・・・・・			・・・・・・	
102	Lib_srs_ptn_open		240			93	ﾊﾟﾀﾝｻｲｽﾞ　250×250 クロスマーク
103	Lib_srs_ptn_regist		4			1	
104	Lib_srs_ptn_save		・・・・・・			・・・・・・	
105	Lib_srs_set_fine_srch_sw		・・・・・・			・・・・・・	
106	Lib_srs_set_speed		・・・・・・			・・・・・・	
107	Lib_srs_srch_conti		次頁			次頁	
108	Lib_srs_srch_exec		次頁			次頁	
109	Lib_stddevi	100.911	4.069	119.303	25.228	5.000	[*1]
110	Lib_xbinary_convert	162.760	10.579	122.396	39.062	6.000	全面

[*1]ＭＭＸ対応（システム Ver1.60以降）です。

濃淡画像ライブラリ実行所要時間代表値一覧

単位　　ミリ秒

項番	ライブラリ名	所要時間					備　　　考
		９０１	９０２	９０３	９０４	FVL/LNX	
111	Lib_xlhough_close	・・・・・・	・・・・・・	・・・・・・	・・・・・・	・・・・・・	
112	Lib_xlhough_detection	１本　56 20本　66	１本　15 20本　15	１本　86 20本　94	１本　30 20本　33	１本　6 20本　6	
113	Lib_xlhough_edge_close	・・・・・・	・・・・・・	・・・・・・	・・・・・・	・・・・・・	
114	Lib_xlhough_edge_open	643	55.338	806	286	43.000	(0,0)-(511,479), エッジ点は15300点　[*1]
115	Lib_xlhough_init_hough_sp	21	9.766	27	8	3.000	Lib_xlhough_openでの 条件に依存する
116	Lib_xlhough_open	１回目 3982 ２回目 8	１回目 178 ２回目 ・・・・・・	１回目 2349 ２回目 6	１回目 1580 ２回目 3	１回目 77 ２回目 3	矩形領域は(0,0)-(511,479), st_q=-179,ed_q=180
117	Lib_xlhough_refine_line	28	2.441	22	7	1.000	エッジ点は15300点
118	Lib_xlhough_support_close	・・・・・・	・・・・・・	・・・・・・	・・・・・・	・・・・・・	なし
119	Lib_xlhough_support_open	26	2.441	63	24	1.200	エッジ点は15300点
120	Lib_xlhough_thres_test	432	52.897	432	278	43.000	処理範囲は(0,0)-(511,479), エッジ点は12366点
121	Lib_xlhough_voting	305	17.090	198	81	10.200	Lib_xlhough_openでの条件 st_q，ed_qに依存するエッジ点は 15300点、vot_wid=10
122	Lib_zero_cross	419	13.021	239.399	79.752	8.400	

[*1]ＭＭＸ対応（システム Ver1.60以降）です。

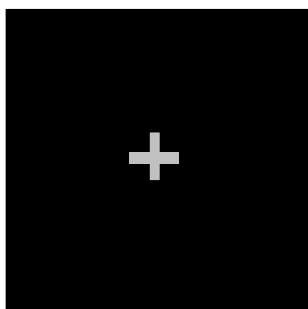
■グレイサーチ処理時間の条件

○ＷＩＮＤＯＷ範囲
　１２８＊１２８（１２８，１２８）－（２５５，２５５）
　２５６＊２５６（１２８，１２８）－（３８３，３８３）
　３８４＊３８４（０６４，０６４）－（４４７，４４７）
　５１２＊４８０（０００，０００）－（５１１，４７９）

○しきい値
　下限　５０００　　　　　　上限　６０００

○処理時間の単位はｍｓ

○９０２は、Pentium 200MHzで測定しています。

○サーチ画像について

左図の画像に対し、中央の十字マークをサーチしました。
また、背景の濃度値は0，十字マークの濃度値は170です。

○ Lib_gs_search

				サーチエリアの大きさ								
				２５６＊２５６			３８４＊３８４			５１２＊４８０		
				通常精度	高精度	超高精度	通常精度	高精度	超高精度	通常精度	高精度	超高精度
パタンの大きさ	32*32	複雑度１	901NT	21	22	25	42	43	46	58	59	62
			902	3	3	3	6	6	6	7	7	7
			903	20	20	23	40	40	43	56	56	59
			904	7	7	7	12	13	14	18	19	20
			FVL/LNX	1	1	1	2	2	2	4	4	4
		複雑度５	901NT	80	80	83	183	183	187	300	300	304
			902	7	7	7	15	15	15	22	23	23
			903	70	72	74	161	162	165	266	266	269
			904	25	26	27	58	59	59	96	96	97
			FVL/LNX	4	4	4	9	9	10	16	16	16
	64*64	複雑度１	901NT	17	20	33	33	35	46	41	43	55
			902	2	2	3	5	5	6	6	6	6
			903	15	17	29	28	30	42	35	38	49
			904	4	5	9	8	9	12	11	11	15
			FVL/LNX	1	1	1	1	1	2	2	2	3
		複雑度５	901NT	29	33	44	63	67	78	94	97	109
			902	3	4	4	7	7	7	9	9	10
			903	26	29	40	55	57	69	83	85	97
			904	8	9	13	18	19	23	28	28	32
			FVL/LNX	1	1	2	3	3	3	5	5	6

				サーチエリアの大きさ								
				２５６＊２５６			３８４＊３８４			５１２＊４８０		
				通常精度	高精度	超高精度	通常精度	高精度	超高精度	通常精度	高精度	超高精度
パタンの大きさ	128*128	複雑度１	901NT	20	31	76	33	45	90	39	50	96
			902	2	3	5	5	5	7	5	6	7
			903	17	28	74	28	39	85	33	45	91
			904	5	8	22	8	11	25	10	13	27
			FVL/LNX	1	1	2	1	1	3	2	2	3
		複雑度５	901NT	20	32	77	37	49	94	48	60	106
			902	3	3	5	5	6	7	6	6	8
			903	17	28	74	32	42	89	42	52	98
			904	5	8	22	10	13	26	12	16	29
			FVL/LNX	1	1	2	1	2	3	2	3	4
	256*256	複雑度１	901NT				46	90	269	51	97	276
			902				5	7	20	6	7	21
			903				39	85	263	45	91	269
			904				11	25	79	13	26	81
			FVL/LNX				1	2	6	2	3	7
		複雑度５	901NT				45	90	269	51	96	275
			902				5	7	20	6	7	20
			903				39	85	263	45	91	269
			904				11	25	79	13	26	80
			FVL/LNX				1	2	6	2	3	7

○ Lib_gs_xsearch (Lib_gs_ysearch)

パタンの大きさ				サーチエリアの大きさ								
				２５６＊２５６			３８４＊３８４			５１２＊４８０		
				通常精度	高精度	超高精度	通常精度	高精度	超高精度	通常精度	高精度	超高精度
	32*32	複雑度１	901NT	7	9	11	16	16	20	25	27	29
			902	1	1	1	2	2	2	2	2	2
			903	8	9	11	17	18	21	28	29	32
			904	2	3	4	7	7	7	11	11	11
			FVL/LNX	0	0	0	1	1	1	1	1	1
		複雑度５	901NT	67	67	70	157	157	160	268	268	271
			902	5	5	5	11	11	11	19	19	20
			903	59	59	62	139	140	142	238	239	242
			904	22	22	23	51	52	53	88	88	89
			FVL/LNX	3	3	3	7	7	7	13	13	13
	64*64	複雑度１	901NT	3	7	18	6	8	20	8	11	23
			902	1	1	2	1	1	2	1	1	2
			903	3	7	17	6	8	20	8	11	22
			904	1	2	6	2	2	7	2	3	7
			FVL/LNX	0	0	0	0	0	0	0	0	0
		複雑度５	901NT	16	19	30	37	39	51	62	65	77
			902	2	2	2	3	3	3	5	5	5
			903	15	17	28	33	36	46	55	59	70
			904	5	6	9	11	12	16	20	20	24
			FVL/LNX	0	1	1	1	2	2	3	3	3

				サーチエリアの大きさ								
				２５６＊２５６			３８４＊３８４			５１２＊４８０		
				通常精度	高精度	超高精度	通常精度	高精度	超高精度	通常精度	高精度	超高精度
パタンの大きさ	128 * 128	複雑度１	901NT	6	17	63	7	18	63	7	19	64
			902	2	2	4	2	2	4	2	2	4
			903	5	16	63	6	17	63	7	17	63
			904	2	5	19	2	5	19	2	5	19
			FVL/LNX	0	0	1	0	0	1	0	0	1
		複雑度５	901NT	7	19	63	11	22	68	16	28	73
			902	2	2	4	2	2	5	2	3	5
			903	6	17	63	9	20	67	14	25	71
			904	2	5	19	3	7	20	5	8	20
			FVL/LNX	0	0	1	0	0	1	0	1	2
	256 * 256	複雑度１	901NT				19	63	243	19	63	243
			902				7	9	24	7	9	24
			903				16	63	240	18	63	241
			904				5	19	72	5	19	72
			FVL/LNX				0	1	5	0	1	5
		複雑度５	901NT				18	63	242	20	64	243
			902				7	9	24	7	9	24
			903				17	63	240	18	63	241
			904				5	19	72	6	19	72
			FVL/LNX				0	1	5	0	1	5

○ Lib_gs_gfreeze

	サーチエリアの大きさ			
	１２８＊１２８	２５６＊２５６	３８４＊３８４	５１２＊４８０
901NT	4.88	14.65	26.86	32.56
９０２	0.814	2.441	4.069	4.883
９０３	4.07	11.39	22.79	26.86
９０４	1.63	3.26	6.51	8.14
FVL/LNX	0.00	0.00	1.00	2.00

○ Lib_gs_pcorr

	パタンの大きさ			
	３２＊３２	６４＊６４	１２８＊１２８	２５６＊２５６
901NT	0.81	1.63	5.70	21.16
９０２	0.00	0.00	0.81	2.44
９０３	0.81	1.63	5.70	20.35
９０４	0.00	0.81	1.63	5.70
FVL/LNX	0.00	0.00	0.00	1.00

○ Lib_gs_ptn_compare

	パタンの大きさ			
	３２＊３２	６４＊６４	１２８＊１２８	２５６＊２５６
901NT	4.07	15.47	59.41	238.45
９０２	0.000	0.814	4.883	18.717
９０３	3.26	10.58	41.50	167.64
９０４	0.81	4.07	17.09	67.55
FVL/LNX	0.00	0.00	0.00	3.00

■８ＢＩＴ版グレイサーチの処理時間

○ Lib_gs_search

				サーチエリアの大きさ								
				２５６＊２５６			３８４＊３８４			５１２＊４８０		
				通常精度	高精度	超高精度	通常精度	高精度	超高精度	通常精度	高精度	超高精度
パタンの大きさ	32 * 32	複雑度１	902	3	3	3	6	6	6	7	7	7
			FVL/LNX	1	1	1	2	2	2	4	4	4
		複雑度５	902	7	7	7	13	14	14	20	20	20
			FVL/LNX	4	4	4	9	9	10	16	16	16
	64 * 64	複雑度１	902	2	2	3	5	5	5	6	6	6
			FVL/LNX	1	1	1	1	1	2	2	2	3
		複雑度５	902	3	4	4	7	7	7	8	9	9
			FVL/LNX	1	1	2	3	3	3	5	5	6
	128 * 128	複雑度１	902	2	3	4	5	5	7	5	6	7
			FVL/LNX	1	1	2	1	2	3	2	2	3
		複雑度５	902	2	3	4	5	5	7	5	6	7
			FVL/LNX	1	1	2	1	2	3	3	3	4
	256 * 256	複雑度１	902				5	6	19	5	7	19
			FVL/LNX				1	2	6	2	3	7
		複雑度５	902				5	6	19	5	7	19
			FVL/LNX				1	2	6	2	3	7

○ Lib_gs_xsearch (Lib_gs_ysearch)

				サーチエリアの大きさ								
				２５６＊２５６			３８４＊３８４			５１２＊４８０		
				通常精度	高精度	超高精度	通常精度	高精度	超高精度	通常精度	高精度	超高精度
パタンの大きさ	32*32	複雑度１	902	2	2	2	2	2	2	4	4	4
			FVL/LNX	0	0	0	1	1	1	1	1	1
		複雑度５	902	5	5	6	11	11	11	20	20	20
			FVL/LNX	3	3	3	7	7	7	13	13	13
	64*64	複雑度１	902	1	2	2	2	2	2	2	2	2
			FVL/LNX	0	0	0	0	0	0	0	0	0
		複雑度５	902	2	2	2	3	3	4	5	5	6
			FVL/LNX	0	1	1	1	2	2	3	3	3
	128*128	複雑度１	902	3	3	5	3	3	5	3	3	5
			FVL/LNX	0	0	1	0	0	1	0	0	1
		複雑度５	902	3	3	5	3	3	5	3	3	5
			FVL/LNX	0	0	1	0	0	1	0	1	2
	256*256	複雑度１	902				12	13	28	12	13	28
			FVL/LNX				0	1	5	0	1	5
		複雑度５	902				12	13	28	12	13	28
			FVL/LNX				0	1	5	0	1	5

○ Lib_gs_gfreeze

	サーチエリアの大きさ			
	１２８＊１２８	２５６＊２５６	３８４＊３８４	５１２＊４８０
９０２	0.814	2.441	4.069	4.883
FVL/LNX	0.000	0.000	1.000	2.000

○ Lib_gs_pcorr

	パタンの大きさ			
	１２８＊１２８	２５６＊２５６	３８４＊３８４	５１２＊４８０
９０２	0.00	0.00	0.81	2.44
FVL/LNX	0.00	0.00	0.00	1.00

○ Lib_gs_ptn_compare

	パタンの大きさ			
	１２８＊１２８	２５６＊２５６	３８４＊３８４	５１２＊４８０
９０２	0.000	1.628	4.883	20.345
FVL/LNX	0.000	0.000	0.000	3.000

■Ｓ回転サーチ実行時間一覧

Lib_srs_srch_exec()の値は下図の画像から十字マークのみを探した場合の処理時間、
Lib_srs_srch_conti() の値は図の状態からLib_srs_srch_exec()で台形のサーチを実行した後に、引き続き Lib_srs_srch_conti()で十字マークサーチを実行した時の処理時間です。
括弧内の値はLib_srs_srch_exec()による台形サーチの時間を表します。

○処理時間の単位はｍｓ

○Lib_srs_srch_exec()

	スケール固定		スケール可変 （縮小パタン）		スケール可変 （拡大パタン）	
	９０２	FVL/LNX	９０２	FVL/LNX	９０２	FVL/LNX
粗＋精サーチ　高速	１０７	４６	１２５	５４	１５７	５３
粗＋精サーチ　普通	１９１	８０	２０５	９３	２４６	８９
粗サーチのみ　高速	６３	３５	５６	４４	１０９	４１
粗サーチのみ　普通	１４４	７１	１６２	８１	１９２	７８

○Lib_srs_srch_conti()（複数パタン一括サーチ）

	スケール固定		スケール可変 （縮小パタン）		スケール可変 （拡大パタン）	
	９０２	FVL/LNX	９０２	FVL/LNX	９０２	FVL/LNX
粗＋精サーチ　高速	50（ 87）	14（39）	68（ 84）	19（42）	90（ 95）	20（39）
粗＋精サーチ　普通	48（167）	13（75）	66（161）	22（79）	89（180）	10（76）
粗サーチのみ　高速	5（ 57）	4（34）	24（ 56）	9（37）	41（ 61）	9（34）
粗サーチのみ　普通	5（136）	4（69）	24（133）	9（74）	42（145）	9（71）

●パタンサイズは十字マークが２５０×２５０、台形が２４０×２００で登録しました。
●スケール可変時についてはスケールの範囲を８０～１２０％で登録し、
　拡大（約１０８％）／縮小（約９３％）の両方についてサーチしています。
●処理時間は、登録パタンや画像の状態によっても変化しますのでご了承ください。

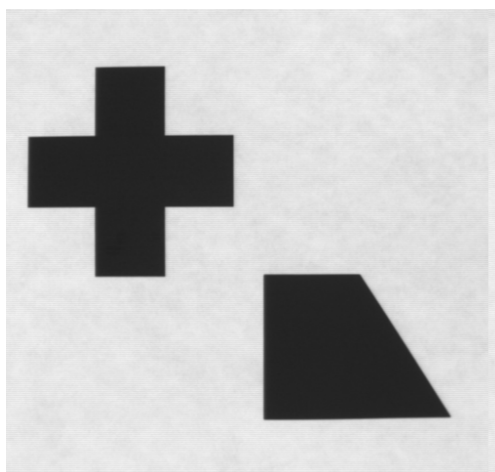



図１．Lib_srs_srch_exec(), Lib_srs_srch_conti()における登録パタンおよびサーチ画像

付録２．正規化相関とは？

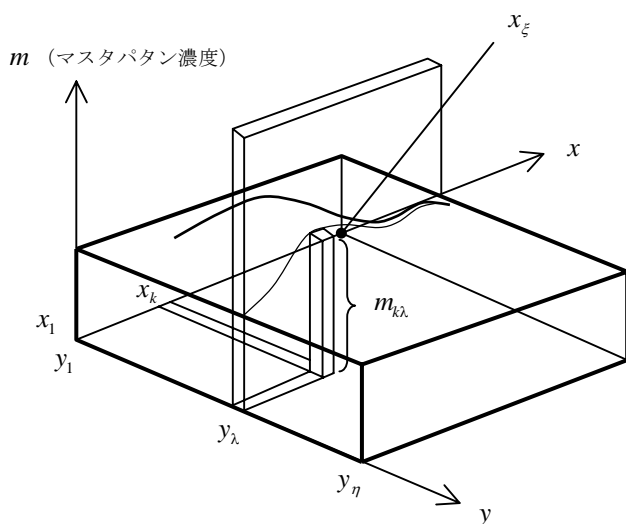



図１　マスタパタンの濃度分布

マスタパタンの濃度分布が図１のようになっているとすると

$y = y_\lambda$

の位置で断面をとると、図２のようになります。

この時、マスタパタンの領域は

$\left(x_1 , y_1 \right) - \left(x_\xi , y_\eta \right)$

の矩形領域として考えています。

また、濃度 $m_{k\lambda}$ は $x = x_k$, $y = y_\lambda$ の位置でのマスタパタンの濃度です。

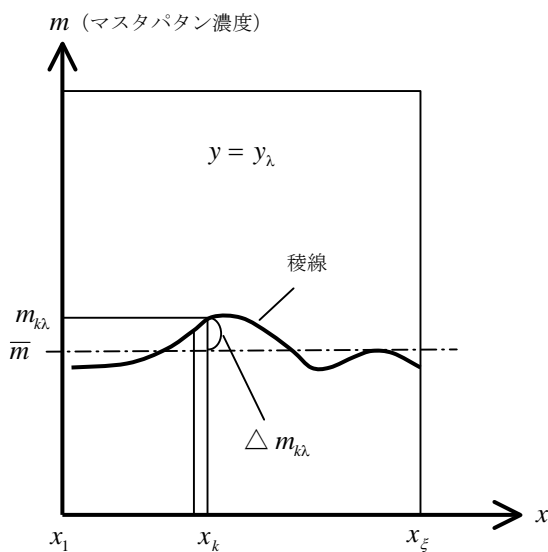



図２　マスタパタンの濃度分布断面

図２において、 $\overline{m}$ は

$$\overline{m} = \sum_{\lambda=1}^{\eta} \sum_{k=1}^{\xi} \frac{m_{k\lambda}}{\xi \times \eta}$$

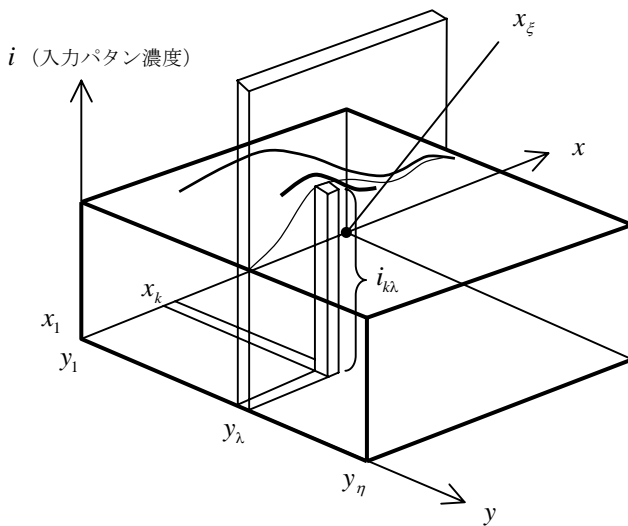
で、つまりはマスタパタンの平均濃度です。

また、 $\triangle m_{k\lambda}$ は

$$\triangle m_{k\lambda} = m_{k\lambda} - \overline{m}$$

で、 $x = x_k$ ， $y = y_\lambda$ の位置における平均濃度からのズレです。

この $\triangle m_{k\lambda}$ を $k = 1$ ， $\lambda = 1$ から $k = \xi$ ， $\lambda = \eta$ まで計算し、並べる事によって、マスタパタンの稜線の形状を表わします。この形状をベクトルに見立てて、 M で表わす事にすれば（1）式のようになります。

$$\mathrm{M} = \left(\triangle m_{11} \quad \triangle m_{12} \ \Lambda\,\Lambda \ \triangle m_{\xi\eta} \right)^T$$ ----------- **(1)**

一方、入力パタンの濃度分布が図３のようになっているとすると

$$y = y_\lambda$$

の位置で断面をとると、図４のようになります。

この時、入力パタンの領域は

$$(x_1, y_1) - (x_\xi, y_\eta)$$

の矩形領域でマスタパタンの領域と同じです。また、濃度 $i_{k\lambda}$ は $x = x_k$, $y = y_\lambda$ の位置での入力パタンの濃度です。

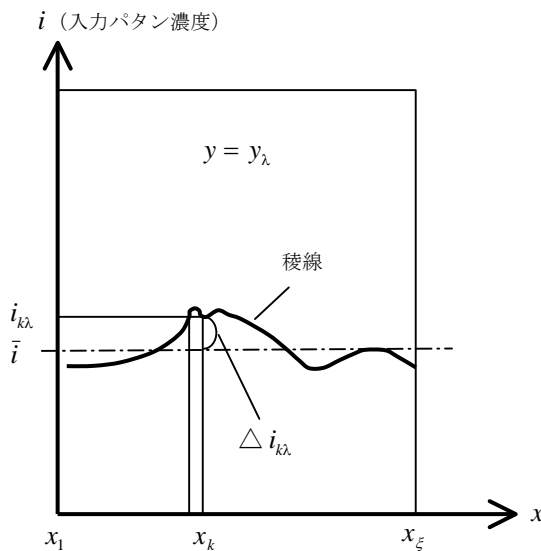



図３　入力パタンの濃度分布

図４において、 $\bar{i}$ は

$$\bar{i} = \sum_{\lambda=1}^{\eta} \sum_{k=1}^{\xi} \frac{i_{k\lambda}}{\xi \times \eta}$$

で、つまりは入力パタンの平均濃度です。

また、 $\triangle i_{k\lambda}$ は

$$\triangle i_{k\lambda} = i_{k\lambda} - \bar{i}$$

で、 $x = x_k$ ， $y = y_\lambda$ の位置における平均濃度からのズレです。

マスタパタンの時と同様に、入力パタンの稜線の形状をベクトルに見立てて、 I で表わす事にすれば（２）式のようになります。

図４　入力パタンの濃度分布断面

$$\mathrm{I} = \left(\triangle i_{11} \quad \triangle i_{12} \ \Lambda\Lambda \ \triangle i_{\xi\eta} \right)^T \quad \text{----------------} \ (2)$$

$\mathbf{M}$, $\mathbf{I}$ とも、各々の平均濃度 $\overline{m}$, $\overline{i}$ を基準とした稜線なので平均濃度分は除去されてしまっている事に注意してください。
つまり、図５の $\mathbf{M}$ と図６の $\mathbf{M}$ は同じものになります。

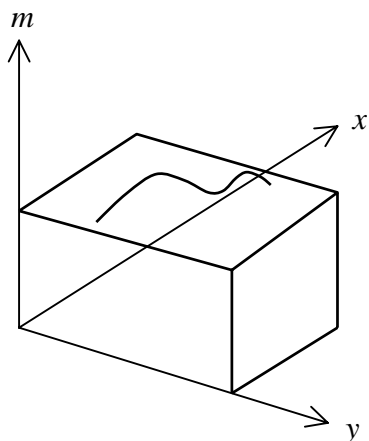



図５　「とある濃度分布」

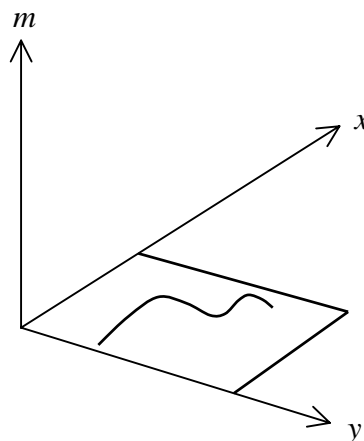



図６　図５とは違う「とある濃度分布」

次に、 $\mathbf{M}$ と $\mathbf{I}$ の類似度を考える事になります。 $\mathbf{M}$ と $\mathbf{I}$ はベクトルで表わしていますので、２つのベクトルの類似度を考えればよい訳です。
数学では、これにピッタリの演算として内積があります。
$\mathbf{M}$, $\mathbf{I}$ は実際は（ $\xi \times \eta$ ）次元上のベクトルですが、これを２次元として説明します。
図７に $\mathbf{M}$ と $\mathbf{I}$ が似ている時のベクトル図を書いています。
図中の θ が０に近い事が判ります。

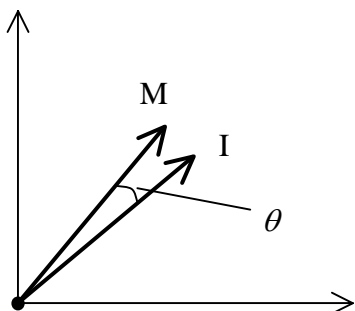



図７　Ｍ と Ｉ が似ている時

図８に Ｍ と Ｉ が似ていない時のベクトル図を書いています。 θ は０からかなり離れます。
図９に Ｍ と Ｉ が最も似ていない時のベクトル図を書いています。 θ はπ（ｒａｄ）になります。
実は、マスタパタンの稜線の形状と、入力パタンの稜線の形状が平均濃度に対して対称となっている時にこうなります。
ある意味では似ているともいえますが…。画像処理の分野では、一般的にこの状態は似ていないと判断しています。
これらの事を踏まえて、内積の定義式を見ることにします。

$$\cos\theta = \frac{\mathrm{M}^T\,\mathrm{I}}{\|\mathrm{M}\|\cdot\|\mathrm{I}\|} \quad \text{-----------------（３）}$$

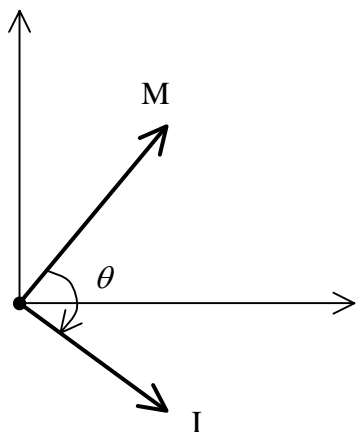



図８　Ｍ と Ｉ が似ていない時

（３）式は内積の定義式そのものですが、 θ は Ｍ と Ｉ のなす角、つまり上述の θ と同じものです。
M^T は Ｍ の転置、 $\mathrm{M}^T\,\mathrm{I}$ は内積、$\|\mathrm{M}\|$は Ｍ の大きさ、$\|\mathrm{I}\|$は Ｉ の大きさ、つまり矢の長さです。（ノルムと言います）
先に述べた通り、似ている時は θ は０に近く、似ていない時は θ は大きくなっていくのですから、 $\cos\theta$ は似ている時は１に近く、似ていない時は０あるいは負値となります。
この $\cos\theta$ を類似度として採用したものが正規化相関です。数学では相関係数と言いますが、同じものと考えてもさしつかえないと思います。つまり、（４）式です。

$$r = \cos\theta = \frac{\mathrm{M}^T\,\mathrm{I}}{\|\mathrm{M}\|\cdot\|\mathrm{I}\|} \quad \text{-----------（４）}$$

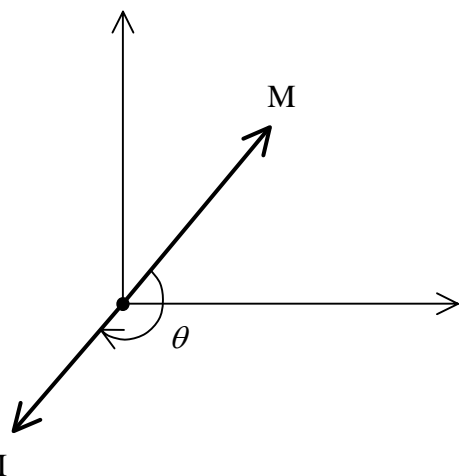



図９　Ｍ と Ｉ が最も似ていない時

正規化という言葉は、（４）式の右辺において、内積を各々のノルムで割っているためです。
（４）式を使うと、入力パタンのコントラストが強くなっても、それに関係なく、一定の値が得られますので、この辺からも画像処理には好都合と言えそうです。なぜ、コントラストに関係なく一定の値が得られるかを簡単に説明します。

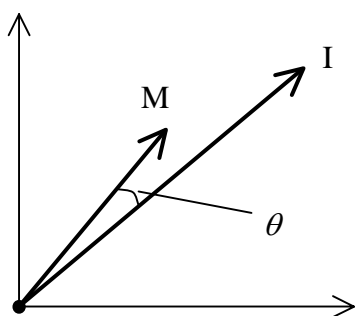



図１０　図７の時よりＩの
コントラストが強くなった時

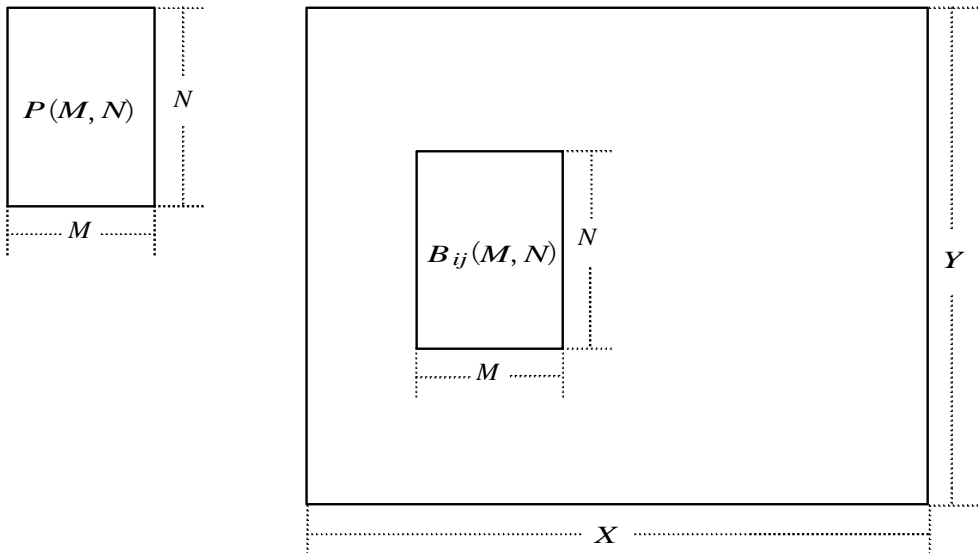
図１０にＩのコントラストが強くなった時のベクトル図を書きました。Ｉの矢が長くなるのみで θ には無関係である事が判ります。
（４）式においても、分子の値が大きくなった比率分、分母の値が大きくなるので、相殺されてしまう訳です。

以上で説明を終えますが、実際の計算は（４）式を具体的に展開したものを使います。
これが（５）式です。（４）式→（５）式の導出は次頁をご覧ください。また、弊社の取扱説明書には（６）式の記載がされていますが、（５）式と全く同等のものですので説明は省略します。
なお、出力としての値は（５）式または（６）式の値を１００００倍したものです。また、負値となった時は強制的に０にしています。

$$r=\frac{n\sum(M_i I_i)-\sum M_i \sum I_i}{\sqrt{n\sum M_i^2-\left(\sum M_i\right)^2}\cdot\sqrt{n\sum I_i^2-\left(\sum I_i\right)^2}} \quad \text{-----------} \ (5)$$

（但し、$\sum \equiv \sum_{i=1}^{n}$ ）

$$C_{ij}=\frac{M\cdot N\cdot\left(\sum_{m=1}^{M}\sum_{n=1}^{N}P(m,n)B_{ij}(m,n)\right)-\left(\sum_{m=1}^{M}\sum_{n=1}^{N}P(m,n)\right)\cdot\left(\sum_{m=1}^{M}\sum_{n=1}^{N}B_{ij}(m,n)\right)}{\sqrt{\left\{M\cdot N\cdot\sum_{m=1}^{M}\sum_{n=1}^{N}P(m,n)^2-\left(\sum_{m=1}^{M}\sum_{n=1}^{N}P(m,n)\right)^2\right\}\cdot\left\{M\cdot N\cdot\sum_{m=1}^{M}\sum_{n=1}^{N}B_{ij}(m,n)^2-\left(\sum_{m=1}^{M}\sum_{n=1}^{N}B_{ij}(m,n)\right)^2\right\}}}$$

この（６）式では、サーチするパタンの画像 $P(M,N)$ と、サーチエリアの画像 $B(X,Y)$ の部分画像 $B_{ij}(M,N)$ との相互相関係数 C_{ij} （マッチングの度合を示す情報）を求めています。

（４）式→（５）式の導出

$$\cos\theta = \frac{\mathrm{M}^T\ \mathrm{I}}{\|\ \mathrm{M}\ \| \cdot \|\ \mathrm{I}\ \|} \quad \text{------------}\ (4)$$

$$\mathrm{M}^T\ \mathrm{I} = \left(\ \triangle m_1\ ,\ \triangle m_2\ ,\Lambda\ \Lambda\ \ ,\ \triangle m_n\ \right)\begin{pmatrix} \triangle i_1 \\ \triangle i_2 \\ \mathrm{M} \\ \mathrm{M} \\ \triangle i_n \end{pmatrix}$$

［注］前頁の通りであると

$$\left(\ \triangle m_{11}\ ,\ \triangle m_{12}\ ,\Lambda\ \Lambda\ \ \triangle m_{\xi\eta}\ \right)\begin{pmatrix} \triangle i_{11} \\ \triangle i_{11} \\ \mathrm{M} \\ \mathrm{M} \\ \triangle i_{\xi\eta} \end{pmatrix}$$

とする所であるが、一般化してこのようにしても式の意味や
効果は全く同じです。

$$= \triangle m_1\ \triangle i_1 + \triangle m_2\ \triangle i_2 + \Lambda\ \Lambda\ \Lambda\ + \triangle m_n\ \triangle i_n$$

$$= \left(\ m_1 - \overline{m}\ \right)\left(\ i_1 - \overline{i}\ \right) + \left(\ m_2 - \overline{m}\ \right)\left(\ i_2 - \overline{i}\ \right) + \Lambda\ \Lambda\ + \left(\ m_n - \overline{m}\ \right)\left(\ i_n - \overline{i}\ \right)$$

$$= m_1\ i_1 - m_1\ \overline{i} - i_1\ \overline{m} + \overline{m}\ \overline{i} + m_2\ i_2 - m_2\ \overline{i} - i_2\ \overline{m} + \overline{m}\ \overline{i} + \Lambda\ \Lambda\ \ m_n\ i_n - m_n\ \overline{i} - i_n\ \overline{m} + \overline{m}\ \overline{i}$$

$$= \sum m_k\ i_k + n\ \overline{m}\ \overline{i} - \overline{i}\ \sum m_k - \overline{m}\sum i_k \qquad （但し、\sum \equiv \sum_{k=1}^{n}\ ）$$

ここで $\overline{m} = \dfrac{\sum m_k}{n}$ ， $\overline{i} = \dfrac{\sum i_k}{n}$ だから

$$\mathrm{M}^T\ \mathrm{I} = \sum m_k\ i_k + n\,\frac{\sum m_k}{n} \cdot \frac{\sum i_k}{n} - \frac{\sum i_k}{n}\sum m_k - \frac{\sum m_k}{n}\sum i_k$$

$$= \sum m_k\ i_k - \frac{1}{n}\sum m_k\ \sum i_k$$

次に

$$\| \mathrm{M} \| \cdot \| \mathrm{I} \| = \sqrt{\triangle m_1{}^2 + \triangle m_2{}^2 + \Lambda\Lambda + \triangle m_n{}^2} \cdot \sqrt{\triangle i_1{}^2 + \triangle i_2{}^2 + \Lambda\Lambda + \triangle i_n{}^2}$$

を変形します。

ここで、 $L_m = \sqrt{\triangle m_1{}^2 + \triangle m_2{}^2 + \Lambda\Lambda + \triangle m_n{}^2}$ とすると

$$\begin{aligned}
L_m &= \sqrt{(m_1 - \overline{m})^2 + (m_2 - \overline{m})^2 + \Lambda\Lambda + (m_n - \overline{m})^2} \\
&= \sqrt{\sum (m_k - \overline{m})^2} \\
&= \sqrt{\sum \left(m_k{}^2 - 2m_k\overline{m} + \overline{m}^2 \right)} \\
&= \sqrt{\sum \left(m_k{}^2 - 2m_k \frac{\sum m_k}{n} + \frac{\left(\sum m_k\right)^2}{n^2} \right)} \\
&= \sqrt{\sum m_k{}^2 - \frac{2}{n}\left(\sum m_k\right)^2 + \frac{1}{n}\left(\sum m_k\right)^2} \\
&= \sqrt{\sum m_k{}^2 - \frac{1}{n}\left(\sum m_k\right)^2}
\end{aligned}$$

同様に、 $L_i = \sqrt{\triangle i_1{}^2 + \triangle i_2{}^2 + \Lambda\Lambda + \triangle i_n{}^2}$ とすると

$$L_i = \sqrt{\sum i_k{}^2 - \frac{1}{n}\left(\sum i_k\right)^2}$$

よって

$$\cos\theta = \frac{\sum m_k i_k - \frac{1}{n}\sum m_k \sum i_k}{\sqrt{\sum m_k{}^2 - \frac{1}{n}\left(\sum m_k\right)^2}\cdot\sqrt{\sum i_k{}^2 - \frac{1}{n}\left(\sum i_k\right)^2}}$$

$$= \frac{n\sum m_k\ i_k - \sum m_k \sum i_k}{\sqrt{n\sum m_k{}^2 - \left(\sum m_k\right)^2}\cdot\sqrt{n\sum i_k{}^2 - \left(\sum i_k\right)^2}}$$

$\cos\theta = r$,　$m_k = M_i$,　$i_k = I_k$ と置き換えて

$$r = \frac{n\sum M_i\ I_i - \sum M_i \sum I_i}{\sqrt{n\sum M_i{}^2 - \left(\sum M_i\right)^2}\cdot\sqrt{n\sum I_i{}^2 - \left(\sum I_i\right)^2}} \quad \text{------- (5)}$$

（但し、$\sum \equiv \sum_{i=1}^{n}$ ）

参　考

相関係数等の詳細な内容に関しては以下の文献を参考にしてください。

1）大村　　平：多変量解析のはなし，日科技連
2）有馬, 石村：多変量解析のはなし，東京文書

索　引

索　　引

R

S

X

Z